Proactive Detection of Performance Problems using Adaptive Thresholds

Neeraj Bhatia, Oracle Corporation

Introduction

Information technology has become an integral part of our lifestyle. It is no more a jaw-drop thing to see many people using a GPS enabled handset to seek ways or someone checking their email, attending meetings while sitting in a flight or someone sending money to his parents living far away. We prefer to do shopping online sitting at our home only (at least I do) rather than standing in long queues. IT has made our lives so simple yet so fast!

As it is said, great benefits come with great prices! And this applies perfectly over IT enabled systems. Now, as we are so habitual to this tech-savvy lifestyle, even a second delay in getting our information makes us mad. We, as end users, want everything with a button click. To cope with these requirements of end users, the technical guys are always on their toes looking around for anything, which can cause a delay in the performance of the system. This leads to constant monitoring of the systems with a detective eye for any red light flashing which would indicate a problem. But as like with everything else, there is a limit to which we can monitor. Certainly, some things come like a tsunami to us and we are forced to just face it.

Imagine you running a website for gifts. Now, as V-day is approaching, you know there will be a huge demand of your gift items and rather than being happy about it, you are more worried about the performance of the website that it should not hang when people would actually need it! Certainly, when we talk about performance, we know some known issues; some problems, which we have already seen and we are ready to track them. How about the workload, which would hit your system all of a sudden? How you should prepare for that? Or worse, you have already enabled some monitoring checks but you are unsure that whether they will actually depict the problem to you or not?

A Real Time Issue

Previously I worked with a Telco organization, which provides telecommunication services across north India region. Most of the Oracle databases were in terabyte range (10-25 TB), highly critical and available on 24 x 7 basis. There were some highly critical business application with daily transactions count around 1 million. Around 2 Gigabytes of data entered into the system daily; 2,000 users simultaneously access the system. For some business applications, response time SLA was less than 1 second. In such competitive environment, it was very important to continuously monitor the system.
We continuously monitored host CPU utilization, uptime, swap utilization, tablespace usage and various business applications by use of SQL scripts. The output of the scripts was pushed to separate dedicated server for alert filtering. There alerts were filtered based on pre-defined criteria such as if CPU utilization crossed 90%, alerts should be sent to level-1 DBA support team, on 95% to level-2 DBA team and so on. After filtering of alerts, these were sent to SMSE (Short Message Service Engine), which then send SMS to the mobile devices. There were caveats with such approach.

**Hard-coded values for thresholds:** Since threshold values are hard-coded fixed values, there may be cases when metric values are just near the thresholds, but you will not be notified since it doesn't touch threshold value yet. For example, if Warning threshold value for Physical reads is set to 1000 per second. System may run near 980 physical reads per second, without raising an alert. Especially for workload metrics, hard-coded threshold values are difficult to determine.

**Cyclic Workload:** Most of the times metrics depend on workload timings. For example, hour of day or day of week. High Physical reads are normal during night batch jobs, during day timings that is not the case.

**Introduction to Oracle Server-generated alerts**

Oracle 10g has offered lots of goodies with its release. A lot of emphasis is given in 10g database to make the life of a DBA better. Its always better to "rip the evil in the bud" so it’s beneficial to be 'proactive" rather than being "reactive". Though mostly all the seasoned DBAs like to keep control in their own hands, but it’s also true that most of the times, the coded scripts, the defined rules don't work for everything. Taking this into consideration, Oracle 10g database has introduced "Alerts" aka "Server Generated Alerts". You may say that what's new in it as it was always there in previous releases too. To answer this, 10g has changed the mechanism completely. In previous releases, EM daemon (EMD) was responsible to generate these alerts. 10g has an independent process assigned for this reactive monitoring, MMON which access Oracle's kernel structures directly thus making Server Generated Alerts a much better option as compared to its cousin, EM alerts.

The mechanism of server-generated alerts depends on the derived metrics. Here is the definition of it from Oracle docs,

*Unit of measurement used to report the health of the system.*

So as the definition says, metrics are used to measure how the database is performing at a certain period of time. There is a metric for almost everything, which an efficient DBA
would like to monitor. Here is a quick pick of some of the metrics, which are available now,

Redo Generated (per second)
User Calls (per second)
Archive Area Used (%)

Oracle database 10g database collects and stores various statistics into the workload repository (AWR). Those statistics are then analyzed to produce various metrics. Oracle database also keeps historical metrics values in workload repository that can be accessed through database views.

**Metrics Elaborated**

Some metrics have associated predefined limiting parameters called thresholds that cause alerts to be triggered when collected metric values exceed these limits. Metrics are computed to raise alarms for two levels, Warning and Critical. We will see different ways to generate these alarms in the course of this paper.

Server-generated alerts can be classified as stateful alerts and stateless alerts. Stateful alerts are threshold-based alerts in the sense that they are raised when the underlined metric value crossed either corresponding warning or critical threshold value. These alerts are automatically cleared when an alert condition clears. On the other hand, stateless alerts correspond to specific database events such as:

- Snapshot Too Old errors
- Recovery Area Low On Free Space
- Recovery Area Space Usage

These are non-threshold-based alerts and go directly to the history table.

MMON wakes up every minute to compute the metric values. In addition, for all the metrics that have thresholds defined, MMON verifies the thresholds and generates the alerts.

It should be noted that for server-generated alerts functionality, STATISTICS_LEVEL initialization parameter should be set to TYPICAL or ALL.

With the introduction of Oracle 10g server generated alerts, performance issues can be monitored automatically out of the box. Since alerts are raised based on the thresholds values being set for a particular metric, it's important to set correct value of thresholds. If the threshold values are not appropriately set and represent the true picture of the workload, the purpose of the alerts mechanism will be lost. While it is easy to set thresholds for metric which represents server capacity limits, for example Host CPU
Utilization (%), it's difficult to find out appropriate values for metrics like Physical Reads (per second) and Response Time (per transaction).

The problem of finding appropriate value of threshold for performance metrics makes the life of DBAs difficult to implement this functionality. In fact this is major justification, why administrators ignore this. By analyzing the behavior of a database bit further, give us a clue that performance metrics varies with workload types, timings and even day of week. One can observe high value of logical reads per second during day time (OLTP behavior) while comparatively higher physical reads during nighttime (BATCH behavior). Even we can observe high metric values during month-end reporting timings, quarter end closing etc. Thus we can conclude that threshold values should adjust according to particular time.

Oracle 10g release 2 introduced a new feature called adaptive threshold that solves this issue. Adaptive thresholds use statistical measures of central tendency and variability to characterize normal system behavior and trigger alerts when observed behavior deviates significantly from the norm. Before explaining further about adaptive threshold functionality, I like to discuss some concepts related to it.

**Metric Baseline**

A metric baseline is a named collection of a target's performance statistics that have been collected at a specific point in time. The underlying assumption is that during comparable workload timings, systems with relatively stable performance reveal similar metric values. Thus during accepted system performance, you can define metric baseline. Once done, Enterprise Manager will calculate performance metrics during the baseline period. Using these values, you can set threshold values for warning and critical thresholds, thus easing your task, which is somewhat difficult to calculate. Metric baselines can be used to implement adaptive thresholds on certain metrics.

Metric baselines are be of two types:

- **Static Baseline**: Static baselines are made up of a single user-defined interval of time that is of particular interest. Some candidate timings for static baseline are weekend reporting timings, end of fiscal year, quarter end reporting timings. These baselines can be used to characterize the workload periods and for comparison against future occurrences of that workload. Let’s suppose you have weekly reporting period every Saturday from 8AM to 4PM. You may like to create static metric baseline for the week, for which you have acceptable performance. You can then use calculated metric statistics as threshold values. Deviation of which will result in alert notifications.

    Enterprise Manager only computes static baseline statistics once, which is when the baseline is created.
• **Moving Window Baseline:** Moving window baselines are defined as some number of days prior to the current date. The number of days for which we can define moving window baseline depends on AWR retention period and possible values are 7, 21, 35 and 91 days. Basically, moving window baseline is a rolling window of specified days (rolling interval) that moves with the current time. Let's suppose you have specified trailing 7 days as a time period while creating moving window baseline. In that case, the most recent 7-day period becomes the baseline period for all metric observations and comparisons today. Tomorrow this reference period drops the oldest day and picks up today.

These baselines allow us to compare current metric values with recently observed history. These are suitable for systems with predictable workload cycles. For example, hybrid systems have higher online transaction processing during daytime, while batch processing, backups and heavy jobs like statistics gathering during late evening till early morning.

Enterprise Manager computes moving window statistics every day, rather than sampling.

**Time Groups**

The workload may have cyclic usage patterns during defined baseline period, such as workload usage during weekends vs. weekdays. An accurately defined time group provides significant metric statistics, which can further used for applying adaptive thresholds.

The supported time grouping schemes are classified in two broad categories and combination of both are possible.

**Daily options: determines how workload usage pattern varies within hours of a day.**

- **By day and night:** Metrics are aggregated within two time groups, day hours (7a.m. to 7p.m.) and night hours (7p.m. to 7a.m.). Such grouping is appropriate for system with performance and usage variations in daytime and nighttime.
- **By hour of day:** Metrics are aggregated within every hour of a day separately. It’s appropriate when there is significant usage variation within hours of a day.
- **None:** It aggregates all hours together. Such a grouping scheme should be used when there is no usage pattern between daytime vs. nighttime or within hours of a day.

**Weekly options: determines how workload usage pattern changes in days of a week.**

- **By day of week:** Metrics are aggregated separately by days of the week. It can be used when there is significant variation in usage within days of week.
- **By Weekdays and Weekends**: Metrics are aggregated in two separate groups depending on day. Metrics from Monday to Friday will be aggregated in Weekday group while Saturday and Sunday in weekend group. Useful to segregate weekdays usage from weekend days.
- **None**: It means metrics from all the days will be aggregated together. It can be used when there is no pattern that is will-fitted in above two grouping policies.

Following table shows the summary of various time grouping schemes:

<table>
<thead>
<tr>
<th>Time Group</th>
<th>Number of Groups</th>
<th>Description</th>
<th>Minimum AWR Retention Period / Time period for the baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>By Day and Night</td>
<td>2</td>
<td>Creates 2 groups:</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· One group consists of Day hours (7a.m. to 7p.m.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Another group consists of Night hours (7p.m. to 7a.m.)</td>
<td></td>
</tr>
<tr>
<td>By Weekdays and Weekend</td>
<td>2</td>
<td>Creates 2 groups:</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· One for Weekdays (Monday through Friday)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Another for Weekends (Saturday and Sunday)</td>
<td></td>
</tr>
<tr>
<td>By Day and Night, over Weekdays and Weekend</td>
<td>4</td>
<td>Creates 4 groups:</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Two groups for Day hours (7a.m. to 7p.m.) over weekdays and weekends</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>· Two groups for Night hours (7p.m. to 7a.m.) over weekdays and weekends</td>
<td></td>
</tr>
<tr>
<td>By Day of Week</td>
<td>7</td>
<td>Creates 7 daily groups</td>
<td>7</td>
</tr>
<tr>
<td>By Day and Night, per Day of Week</td>
<td>14</td>
<td>Creates 14 groups:</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· 7 groups for Day hours (7a.m. to 7p.m.) for seven days</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>· 7 groups for Night hours (7p.m. to 7a.m.) for seven days</td>
<td></td>
</tr>
<tr>
<td>By Hour of Day</td>
<td>24</td>
<td>Creates 24 hourly groups</td>
<td>21</td>
</tr>
<tr>
<td>By Hour of Day, over Weekdays and Weekend</td>
<td>48</td>
<td>Creates 48 groups:</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· 24 hourly groups for weekdays</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>· 24 hourly groups for weekends</td>
<td></td>
</tr>
<tr>
<td>By Hour of Day, per Day of Week</td>
<td>168</td>
<td>Creates 168 groups:</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· 120 groups for weekdays</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>· 48 groups for weekends</td>
<td></td>
</tr>
</tbody>
</table>

As an example of time grouping schemes, let’s suppose you have an OLTP environment, where online users connect during daytime (call centre employees). It has been observed that 11a.m. to 2p.m. and 5p.m. to 7p.m. during weekdays are peak usage hours. Daily during early hours, backups are scheduled. Also, on weekends, system is mainly used for MIS reporting, maintenance (purging, statistics gathering etc). In such a workload pattern, it is recommended to have metric baseline with “By hour of day over Weekdays and Weekends” time grouping.

**Adaptive Thresholds Methodologies**

Once Metric Baselines period and time groups are defined, we can set adaptive thresholds for some metrics. Adaptive thresholds automatically adjust Critical and Warning threshold values for normal fluctuations within a time group. For a candidate metric you can set adaptive thresholds in one of two possible methodologies: Significance level and Percentage of maximum.
Significance Level Thresholds: Thresholds based on significance level use statistical relevance to determine which current values are unusual. Thresholds are percentile based and four level are possible: High, Very High, Severe and Extreme. In simple terms, if the significance level is set to .99 for a critical threshold, the threshold is set where 1% of the baseline values fall outside this value and any current values that exceed this value trigger an alert. A higher significance level of .999 or .9999 causes fewer alerts to be triggered.

Following table list four available significance levels are their meaning:

<table>
<thead>
<tr>
<th>Significance Level</th>
<th>Based-on Percentile</th>
<th>No. of Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme</td>
<td>0.9999</td>
<td>1 out of 10,000 occurrences</td>
</tr>
<tr>
<td>Severe</td>
<td>0.99</td>
<td>1 out of 1,000 occurrences</td>
</tr>
<tr>
<td>Very High</td>
<td>0.99</td>
<td>1 out of 100 occurrences</td>
</tr>
<tr>
<td>High</td>
<td>0.95</td>
<td>5 out of 100 occurrences</td>
</tr>
</tbody>
</table>

Percentage of Maximum Thresholds: These types of thresholds computes the threshold values based on specified percentages of the trimmed maximum observed over the time group you selected. A trimmed maximum is a representative higher value after outliers (exceptionally high values) have been eliminated. To get a trimmed maximum, Oracle selects 99th Percentile of the maximum value observed, thus eliminate the distortions. Percentage-of-maximum-based alerts are generated if the observation is at or above the percentage of maximum you specify.

For instance, if the trimmed maximum during a time group is 1000, and if 105 is specified as the Warning level, then values above 1050 (That is, 105% of 1000 = 1050) during the same time group will raise an alert.

For both types of alerts you can set the Occurrences parameter, which indicates how many consecutive occurrences must happen before generating the alert.

Eligible Metrics

There are more than 130 metrics that are available in Oracle database 10gR2 but only 15 metrics on which you can set adaptive thresholds. These metrics are grouped into following three categories.

- Performance Metrics: These three metrics indicate current level of system performance.
- Workload Type Metrics: There are four metrics which describe the workload type.

- Workload Volume Metrics: These are eight metrics which indicate how much work is being performed by the system.

<table>
<thead>
<tr>
<th>Metric Type</th>
<th>Metric Name</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Metrics</td>
<td>Database Time</td>
<td>centiseconds per second</td>
<td>The amount of time the database is spending doing actual work</td>
</tr>
<tr>
<td></td>
<td>Response Time</td>
<td>per transaction</td>
<td>How much time it takes to complete each logical transaction</td>
</tr>
<tr>
<td></td>
<td>System Response Time</td>
<td>centi-seconds</td>
<td>How the system is responding to requests</td>
</tr>
<tr>
<td>Workload Type Metrics</td>
<td>Database Block Changes</td>
<td>per transaction</td>
<td>Total number of calls (user and recursive) that executed SQL statements</td>
</tr>
<tr>
<td></td>
<td>Enqueue Requests</td>
<td>per transaction</td>
<td>Amount of redo generated in bytes</td>
</tr>
<tr>
<td></td>
<td>Total Parses</td>
<td>per transaction</td>
<td>Total number of parse calls (hard and soft)</td>
</tr>
<tr>
<td></td>
<td>Session Logical Reads</td>
<td>per transaction</td>
<td>This includes logical reads of database blocks from either the buffer cache or process private memory</td>
</tr>
<tr>
<td>Workload Volume Metrics</td>
<td>Executes</td>
<td>per second</td>
<td>Number of calls (user and recursive) that executed SQL statements</td>
</tr>
<tr>
<td></td>
<td>Redo Generated</td>
<td>per second</td>
<td>Amount of redo generated in bytes</td>
</tr>
<tr>
<td></td>
<td>Network Bytes</td>
<td>per second</td>
<td>Number of bytes transmitted across the network</td>
</tr>
<tr>
<td></td>
<td>Physical Writes</td>
<td>per second</td>
<td>Total number of data blocks written to disk</td>
</tr>
<tr>
<td></td>
<td>Physical Reads</td>
<td>per second</td>
<td>Total number of data blocks read from disk</td>
</tr>
<tr>
<td></td>
<td>Current Logons</td>
<td>Count</td>
<td>Total number of current logons</td>
</tr>
<tr>
<td></td>
<td>User Calls</td>
<td>per second</td>
<td>Number of user calls such as login, parse, fetch, or execute</td>
</tr>
<tr>
<td></td>
<td>Number of Transactions</td>
<td>per second</td>
<td>Number of transactions processed</td>
</tr>
</tbody>
</table>

After understanding the basic concepts, let’s dive into implementation details.

**Creating Metric Baseline**

Before creating metric baselines, you have to activate the metric baseline feature, which is disabled by default. In order to enable it, select database instance Performance tab and click on Metric Baselines link under Related Links section. Click on Enable Metric Baselines button, to enable metric baselines.

Once Metric Baselines is enabled, you’ll get following page.
Here you’ll get options to create Moving window baseline or Static metric baseline. You can also make either metric baseline active. It is worth to note that only one type of baseline can be active at a time.

In order to create moving window baseline, we need to select trailing number of days and appropriate time grouping scheme. This page give us an option to directly set adaptive thresholds for already created metric baseline.

Let’s take an example to create static metric baseline. Click on Manage Static Metric Baseline link under Related Links section. On the Manage Static Metric Baselines page click on create button.

This will give you Create Static Metric Baseline Page. Here you need to specify Metric Baseline Name, Start date and End date of the time period for which you want to create baseline. It must be at least 7 days long. This time period should represent stable system performance. Adaptive thresholds during stable system performance will help comparing metric values in future.

Under Time grouping section, you can specify time grouping scheme (day and week) to adjust baseline for cyclic day or week usage patterns.

Select the Statistics Preview section and click on Compute Statistics button, It will facilitate to determine if there is sufficient data to configure basic statistical alert thresholds. Once the Statistical preview generation completed, Oracle will mark a metric with Warning symbol for which there is not sufficient data to configure adaptive
thresholds. If there is insufficient data, extend the time period or make time groups larger to aggregate statistics across larger data samples, for example select Weekdays and Weekend grouping scheme instead of Day of Week.

Distribution of particular metric values across selected time groups can be viewed by clicking the Details icon.

Click OK button to create static metric baseline.

Setting Adaptive Threshold

Once completed, you will get metric baseline name in the list and Set Adaptive Thresholds button will be enabled. Just click on it and you will get to Manage Adaptive Thresholds page. Here you can set adaptive thresholds for 15 metrics that we have discussed earlier. Clicking on eyeglass icon for a metric will show you distribution of metric value across baseline time period.

In order to set adaptive thresholds, check a metric name and click on Edit button.

As an example, I have selected Physical Reads (per second) metric to set threshold. Next Page will give you list of metric names, you have selected for setting adaptive thresholds and give you an option to select threshold type: threshold based on Significance Level or Percentage of Maximum.
Once thresholds are set for a metric, threshold type will be changed from None.

**Monitoring Metric Statistics**

For each of three categories of baseline metrics (Performance Metrics, Workload Volume Metrics, and Workload Type Metrics), you can view statistical significant change through Baseline Normalized Metrics Page. You can get a link for this on the bottom of metric Baselines page.

Baseline Normalized Metrics Page displays graph for three metrics from each category. The name and position of these metrics can be selected by clicking on Configure button. Thus you can select a combination of nine metric.

Baseline Normalized graph shows a normalized view of metric data. By normalizing we mean that changing it to a known range so it can be compared to other metrics computed in different units (or differing by scale). Here, the metrics are normalized by significance level so that different metrics, possibly having widely varying absolute values will be charted at the same height when their observed values have the same respective statistical significance.
For all the metrics which you’ve configured for adaptive thresholds, alerts will be displayed in Alerts section on database Instance home page. You can get details of alerts by clicking on the alert message link. This will show you real time as well as historical statistics for the metric.
You can get historical alert details of particular metric from Alert History link from Related Links section on database instance home page.

Oracle database 11g Enhancements

Oracle database 11g further enhance the concept of metric baselines and simplify the management of baseline metric thresholds. In this section we’ll briefly discuss the 11g enhancements.

AWR Baseline

Oracle Database 11g introduced the term AWR baseline which is similar to Metric Baseline available in previous release.

Now there are three types of AWR baselines: Single (or static in previous release), Moving Window baseline and Repeating. Repeating AWR baseline are introduced in 11g. These repeat over a time period. For example, every Monday from 10:00 AM to 2:00 PM for the year 2009. Creation of Repeating AWR baseline is possible with the help of Baseline templates. Baseline templates enable us to schedule the creation of baselines for any time period of interest in future. Inputs to the baseline templates are Baseline
name prefix, duration, frequency, Start time, End time and retention. MMON background process periodically checks for the relevant time periods and automatically creates the baseline. Thus creation of AWR baseline gets automated.

**Inbuilt Moving Window Baseline**

An inbuilt Moving window baseline ships with the database and known as SYSTEM_MOVING_WINDOW. It corresponds to last eight days of AWR data. In release 11g, adaptive thresholds functionality computes statistics on this baseline.

**Baseline Display**

The data for any defined baseline in the past is available on the Performance page of Enterprise Manager.

In the page given below, system moving window baseline is used to compare current metric values with the baseline values (99th percentile).
Easy Configuration of Baseline Metric Thresholds

In newer release, EM offers quick configuration of metric thresholds where one can set threshold values in a single mouse-click. There are three possible workload profile, which have different set of metrics.

- Primarily OLTP (pure transaction processing 24 hours a day)
  - Average Active Sessions
  - Redo Generated (per second)
  - Response Time (per transaction)
  - Session Logical Reads (per transaction)

- Primarily Data Warehousing (query and load intensive)
  - Average Active Sessions
  - Cumulative Logons (per second)
  - Physical Reads (per second)
  - Response Time (centi-seconds per call)

- Alternating (OLTP during the daytime and batch during the nighttime)
  - Average Active Sessions
  - Redo Generated (per second)
  - Session Logical Reads (per transaction)
  - User Calls (per second)

Conclusion

So, as we have seen, there were lots of challenges with the traditional monitoring systems and methodologies. With Oracle Database 10g's innovative step of Server Generated Alerts and adaptive thresholds, a lot of the burden over DBA's shoulders is relieved now. Rather than spending time with having a keen eye over the systems to check whether they are working optimally or not, using this powerful technique, they can safely assume that before something would go wrong, alerts would be there to monitor it, making DBA more eligible to do corrective measures way before an issue becomes an error.

About the Author

Neeraj Bhatia is a Senior Technical Analyst in Oracle India, based in Noida, India. He has been working with Oracle Databases for 5 years. Currently he is responsible for Capacity Planning for Oracle-based systems that include Oracle Database, Oracle Application Server, and Oracle Applications. Prior to this, he worked as a Performance Analyst for VLDBs. When not working with Oracle, he likes to listen music, watch movies and spend a good time with family. He can be reached at: neeraj.dba@gmail.com