METRIC BASELINES:

DETECTING UNUSUAL PERFORMANCE EVENTS USING SYSTEM-LEVEL METRICS IN EM 10GR2

John Beresniewicz
Oracle Corporation

ORACLE

INTRODUCTION

This paper concerns the metric baselines feature introduced in Enterprise Manager 10gR2. Metric baselines are simply statistical characterizations of system performance over well-defined time periods. We use metric baselines to implement adaptive alert thresholds for certain performance metrics as well as to provide normalized views of system performance. Adaptive alert thresholds are used to detect and inform system administrators about unusual performance events. Baseline normalized views of metric behavior help administrators explain, understand and diagnose such events.

SYSTEM PERFORMANCE

Enterprise systems are ever larger and supporting ever more users. Applications with thousands of objects and hundreds of thousands of SQL statements are not uncommon. Internet systems may support thousands of concurrent users, with the potential for huge demand spikes always a possibility.

We are interested in looking at large systems and large complexes of systems at the macro-level through metrics that characterize performance and workload of the entire system in some meaningful way. In the case of Oracle RDBMS systems metric values are aggregated across many SQL statements, users or transactions. In the case of web services a metric may represent the composite result or aggregate value summed across a chain of connected subcomponents. In both cases we argue that statistical techniques can be valuable in characterizing and analyzing historical metric observations for comparison purposes with current, past or future system states. We call this process of comparing system behavior (as observed through metrics) to a well-defined characterization of previously observed system behavior *baselining the system*. Statistical baselines allow us to evaluate some measure of how unusual an observed metric value is based on the distribution of previous observations. We expect that statistically unusual observations in certain metrics may signal unusual system states that could be of interest to administrators.

Alert thresholds

Metrics are typically compared to threshold values for purposes of detecting problems and signaling alerts to administrators. Many products, including previous versions of Enterprise Manager, allow for static thresholds manually configured by users. Unfortunately, there are few if any performance metrics for which meaningful thresholds can be established independent of the system to which they will be applied. Threshold configuration under these circumstances can be time consuming, difficult and is often based on unsound analysis or poor data.

Enterprise Manager 10gR2 metric baselines enable alert thresholds to be determined by well-defined statistical computations over time series of metric values previously observed on the system. Thresholds can be set to alert on statistically unusual observations for the system in question, based on its own prior history. These deterministic threshold definitions have semantic power completely absent in the arbitrariness of fixed thresholds.

Workload periodicity

Many systems have usage or workload profiles with cyclic patterns over time. For instance, a corporate email system will exhibit usage patterns closely aligned with employee working schedules. We expect the system to be used most heavily for individual email transactions during the normal working hours, while in the evening perhaps large batch or ETL jobs push data around. The number of users is consistent from day to day, perhaps slowly and steadily rising as the company workforce grows.

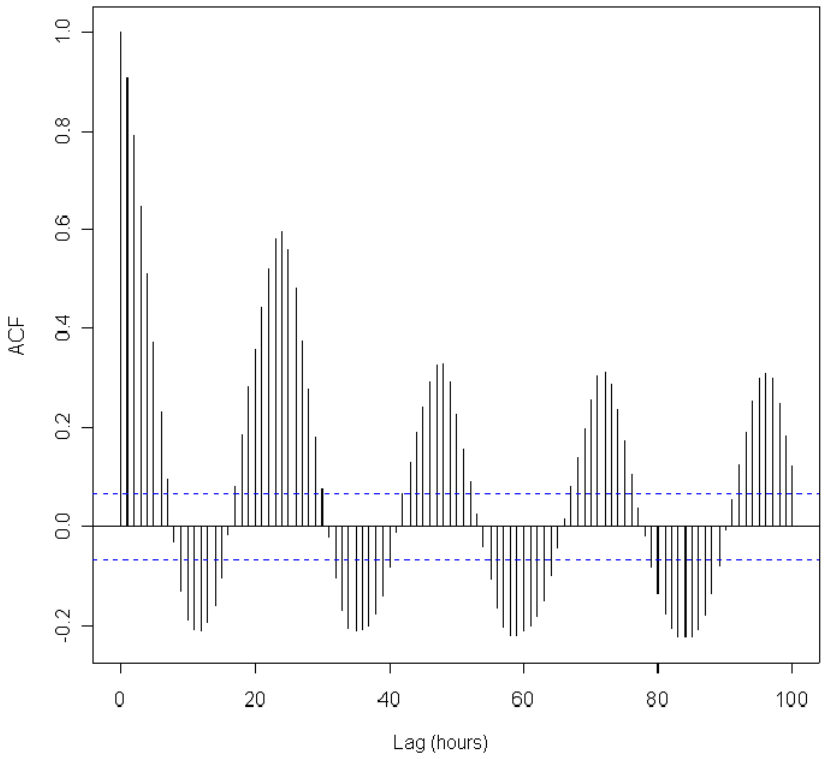

Figure 1: Autocorrelation function of metric *user calls per second* on mail database

Figure 1 is a beautifully symmetric graph. It shows the autocorrelation function of *user calls per second* on an internal mail database for lag times measured in hours. We see that the highest autocorrelation for this demand metric occurs at 24 hours, with other peaks at multiples of 24 hours. Observe also the significant negative autocorrelation at 12-hour intervals. Thus we conclude that demand on the email system has strong diurnal characteristics: *user calls per second* at 9am today is very much like *user calls per second* either yesterday or tomorrow at 9am, and very much unlike *user calls per second* at 9pm.

Workload periodicity presents a serious problem for fixed alert thresholds, as different workloads may have completely different performance characteristics and thus expected ranges of metric values. Different thresholds should apply for these different workloads, and the user should not be responsible for making these changes. The adaptive thresholds component of the EM 10gR2 metric baselines feature addresses this problem directly for common daily and weekly workload periodicities.

Performance events

An important underlying assumption of metric baselines is that systems with relatively stable performance should exhibit similar stability in metric observations over times of comparable workload. However, even in the most closely observed and tuned systems there can be significant variations in performance up to and including loss of service levels (*"things happen…"*). We expect to find that unexpected performance events in large stable systems will be visible as statistically unusual sequences of metric observations. The metric baselines feature includes a display of metric time series that visually highlights sequences of statistically unusual observations. We hope this will prove useful to better understanding complex system dynamics, especially at or near the limits of acceptable performance.

Metric categories

In an effort to better understand and quantify aggregate system behavior using metrics we classify them into broad categories. The following important metric categories are among those recognized in EM 10gR2:

- FAULT
- PERFORMANCE
- WORKLOAD VOLUME
- WORKLOAD TYPE
- CAPACITY

Metric baselines in EM 10gR2 are implemented for a select subset of metrics from the performance, workload volume, and workload type categories. These categories include metrics for which fixed thresholds can be problematic and that have been observed to exhibit heavy-tailed non-normal statistical distributions.

Metrics of fundamental interest are those in the PERFORMANCE category. Performance metrics typically measure response time or some close relative. These metrics should tell us something about how well the system is performing from an end-user perspective.

System workload is broken into two categories of metrics. Workload VOLUME metrics measure load or demand on the system, or the amount of work tasked to the system. Workload TYPE metrics attempt to distinguish the kind or shape of the work tasked to the system.

10gR2 metrics

In Oracle 10gR2 there are 15 of the many metrics visible in V$SYSMETRIC_HISTORY that can be persisted to AWR snapshots as a group by setting a special initialization parameter. These persisted metrics, found in DBA_HIST_SYSMETRIC_HISTORY, are used as raw input data for statistical computations of the metric baselines feature.

The following table lists the metrics available for baselining in RDBMS 10gR2 organized by category.

Metric_ID	Performance Metrics
2106	SQL Service Response Time
2109	Response Time Per Txn
2123	Database Time Per Sec
	Workload Volume Metrics
2003	User Transaction Per Sec
2004	Physical Reads Per Sec
2006	Physical Writes Per Sec
2016	Redo Generated Per Sec
2026	User Calls Per Sec
2058	Network Traffic Volume Per Sec
2103	Current Logons Count
2121	Executions Per Sec
	Workload Type Metrics
2031	Logical Reads Per Txn
2045	Total Parse Count Per Txn

2066	Enqueue Requests Per Txn
2072	DB Block Changes Per Txn

These metrics have been singled out from the more than 100 available metrics in Oracle 10g as a reasonable set of system-level indicators of performance, workload demand and workload shape. It is likely that different metrics will be better indicators on specific systems depending on application characteristics.

METRIC DATA ANALYSIS

We undertook a study of actual metric data harvested from an Oracle internal email database over a period of approximately 2 months. Over 30 metrics were collected and analyzed. The database was running Oracle 10gR1 and data was selected periodically out of V$SYSMETRIC_HISTORY.

Raw data characteristics

In Figure 2 below we see a plot of one week (Monday to Sunday) of raw data for the workload volume metric *executions per second*. We first observe that the plot is not smooth; the data jumps up and down very noticeably. However there is also distinct regularity and patterning in the data. For instance, there are 5 wavy patterns in the heart of the data that are each apparently one day in duration. This is the diurnal pattern noticed in the autocorrelation plot. The weekend days are not as regular and significantly different than the weekdays. Finally, variance in the data is much more pronounced on the high-value side where we occasionally observe what appear to be extreme spikes.

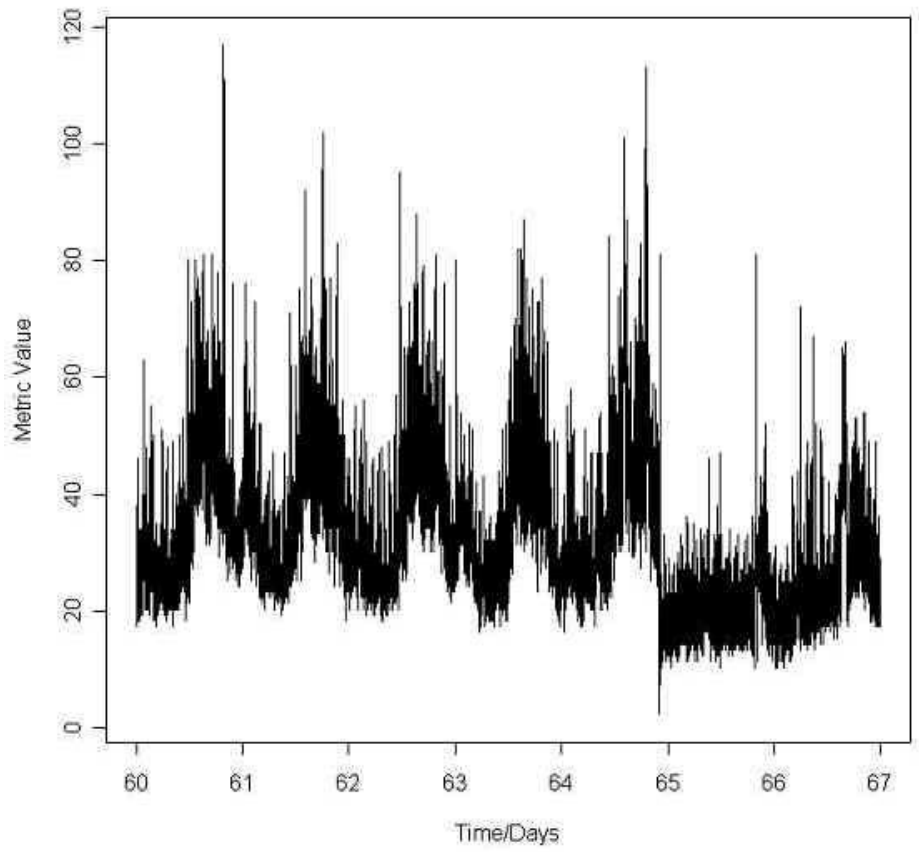

Figure 2: One week of raw data for metric *executions per second*

In summary the raw data for this metric exhibit noisy-ness at the level of individual data points but some regularity at the aggregate level. There appears to be common up-and-down patterning within weekdays as well as a distinct break between weekends and weekdays. It should also be evident that defining a single threshold value on this metric for alerting will be very difficult.

Metric value distributions

We investigated the common aggregate properties of metric values by first analyzing histograms of their distributions. The data was separated by hour of the day and initially only weekdays were analyzed as they showed the strongest daily patterning as seen previously in Figure 2.

Figure 3 below shows one such histogram for values of the metric *executions per second* collected between 4pm and 5pm.

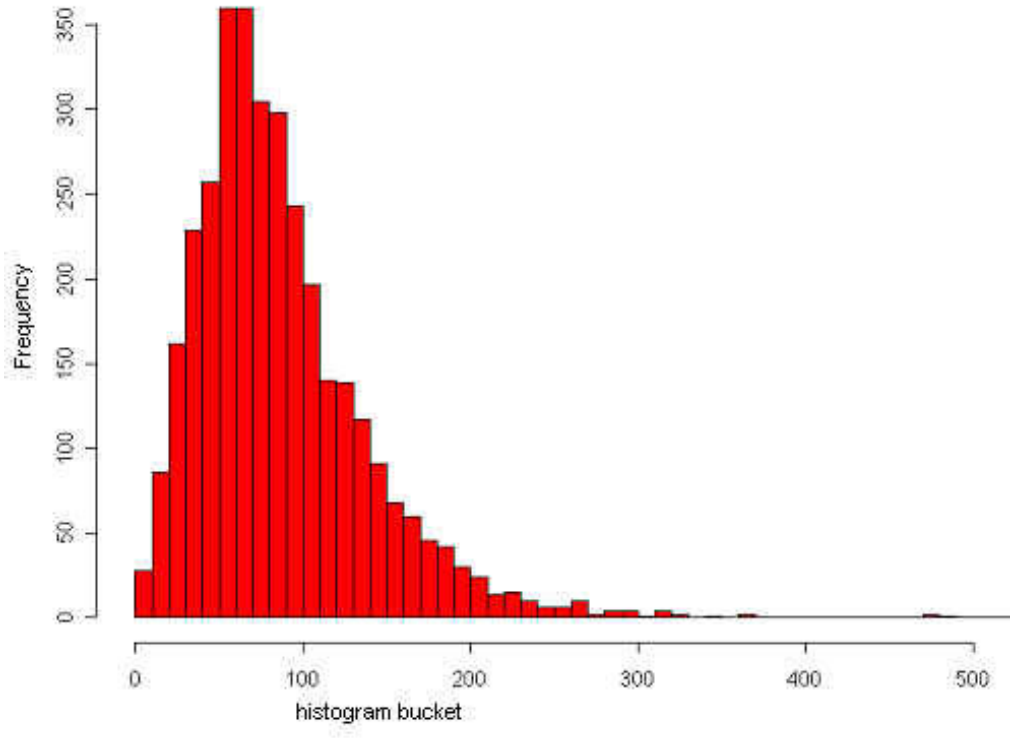

Figure 3: Histogram of *executions per second* for 4pm hour

Notice that the distribution does not have the classic "bell-shaped curve" of the normal distribution. The higher values tail off much more slowly than the lower ones, and much more slowly than tails of the normal distribution. In fact this particular example looks very much like a gamma distribution of data.

We observed similar histograms across many metrics. In some cases the bulk (or hump) of the distribution was not so regularly shaped as Figure 3, but in almost every case there was this heavy-tailed decline on the high side.

Recall that we are interested in identifying unusual metric values, which is to say statistically unlikely observations. This requires that we somehow model the statistical distribution of metric values from prior observations with special attention to the tails, as that is precisely where low probability values are to be found. The problem is to identify the likelihood of an observed high value given that we do not have perfect knowledge of the true distribution. For instance, in Figure 3 we see at least one observation close to the value 500. What is the probability of making this observation? If it is 1-in-100 then we might think it not so unusual, whereas if it is 1-in-10000 then perhaps we consider it highly unusual.

Exponential tail modeling

The exponential distribution is quite commonly used for various parameters in queuing models of computing systems (e.g. arrival rates and service times.) Exponential distributions also have the high-side heavy-tail property observed in our data. We decided to use the family of exponential distributions in statistically modeling our observed data.

Since we are most interested in modeling tails of metric distributions (for purposes of identifying the unusual) we do not concern ourselves so much with the "hump" of the data and focus more on high value observations. This approach introduces two issues. First, we require a larger total number of observations to obtain sufficient statistical strength as our model is based only on a subset of all observations. Second, the model could be corrupted by occurrences of the abnormally high values we are trying to identify in the first place. That is, the model could be "polluted" by outliers.

One approach to fitting observed data to an exponential tail model is illustrated in Figure 4 below. We compute a specific transform function on the data such that exponentially distributed data will be linear in the transform space. Next we perform linear regression on a subset of the high-end of observed values that excludes possible outliers. This provides the exponential model parameters, which are then used to estimate values for high significance levels, e.g. 0.999 and 0.9999 in Figure 4.

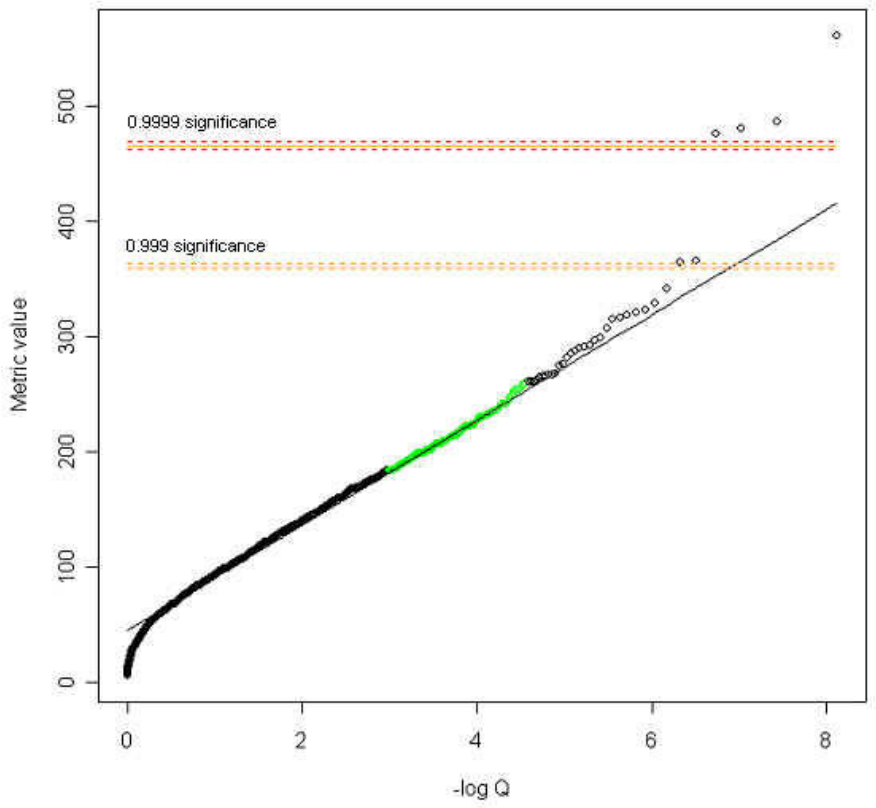

Figure 4: Exponential tail estimation of *executions per second* for 4pm hour

Finally we can use these high significance estimates based on the derived model parameters as thresholds for the detection of statistically unusual observations. In Figure 4 we see that four observations actually lie above the four-nines (0.9999 or 1-in-10000) level of significance according to the model and may represent unusual system activity about which the administrator should have been made aware.

METRIC BASELINES

The metric baselines feature is designed to provide a stronger basis for system or service level analysis and comparison of macroscopic performance indicators captured as metrics. In particular, statistical distributions of metric values are characterized using measured percentiles for the bulk of the distribution and computing parameters for estimating high tail percentiles using exponential assumptions about the tail. These estimated high percentile values are used to identify outlier metric readings for purposes of signaling unusual system behavior.

Baseline period

We define baselines as sets of well-defined time intervals over which we have captured system performance metrics for purposes of comparison with current, past or future states of the system.

Enterprise Manager 10gR2 baseline periods are limited to a single time interval. The broader concept of baseline period specifically includes the possibility of multiple disjoint intervals as constituting a period.

There are two types of baseline period supported in 10gR2:

- Moving window baseline period
- Static baseline periods

There will likely be some confusion with baseline periods and time partitions as defined below. We apologize in advance for this and any other confusing concepts or ambiguous terms.

Moving window baselines

The moving window baseline period is defined as some number of whole days prior to the current date. The period is moving in the sense that as time moves forward, so does the baseline period. Baselining systems using moving window periods allows us to compare current metric values with recently observed history. Thus, the moving window baseline adapts to system evolution by incorporating new data into computations. Moving window baselines should be most useful for large operational systems with strong and predictable workload cycles (e.g. OLTP days and batch nights).

Static baselines

Static baselines are named periods of time defined by and of specific interest to customer environments. Static baselines can be used to capture and characterize specific workload processing periods after they have occurred for comparison against future occurrences of that workload.

Time partitioning

The baseline period can be divided into sub-intervals of time over which statistical aggregates are computed. These sub-intervals, or “time partitions” are intended to allow baselines to capture and adapt thresholds to expected time-dependent workload variations.

EM 10gR2 allows for simple time partitioning that can capture common daily or weekly usage/workload cycles. The daily options are:

- By hour of day: aggregate each hour separately, strong variation across hours
- By day and night: aggregate the hours of 7am-7pm as day and 7pm-7am as night
- By all hours: aggregate all hours together, no strong daily cycle

The weekly time partitioning options are:

- By day of week: aggregate days separately, strong variation across days
- By weekday and weekend: aggregate Mon-Fri together and Sat-Sun together

- By all days: aggregate all days together, no strong weekly cycle

Time partitioning is fully specified in EM 10gR2 by selecting both daily and weekly partitioning options.

For example, consider an operational system serving many online users working similar daily schedules and running batch jobs in the evening. In this case, the following time partitioning is a reasonable choice:

```
By day/night AND by weekday/weekend
```

Note the relationships between the baseline period, granularity of time partitioning and amount of data in each aggregate:

- Finer time partitioning => less data per aggregation
- Larger baseline period => more data per aggregation

The aggregate cardinality is important in that statistical strength of the percentiles and especially estimated values depends in part on having sufficient data. It is not intended that customers be required to understand this level of detail in order to successfully use the feature.

Active baseline

In EM 10gR2 there can be a single moving window baseline and as many static baselines as users define. There can be at most one baseline designated as active on the target. The active baseline primarily controls the setting of adaptive alert thresholds.

BASELINE STATISTICS

The core of the metric baselines feature is the statistical profile of metric values over the baseline period. These statistics are computed at baseline creation for static baselines and once per day (ideally around 12 midnight) for moving window baselines.

Computed values

The following object type definition constitutes the statistical aggregate computed over each time partition of the baseline period.

```
create type bsln_statistics_t as object
   (bsln_guid          raw(16)
   ,datasource_guid    raw(16)
   ,compute_date       date
   ,subinterval_code   raw(21)
   ,sample_count       number
   ,average            number
   ,minimum            number
   ,maximum            number
   ,sdev               number
   ,pctile_25          number
   ,pctile_50          number
   ,pctile_75          number
   ,pctile_90          number
   ,pctile_95          number
   ,est_sample_count   number
   ,est_slope          number
   ,est_intercept      number
   ,est_fit_quality    number
   ,est_pctile_99      number
   ,est_pctile_999     number
   ,est_pctile_9999    number
   );
```

The statistical distribution of the metric values over the baseline period is captured using measured percentiles up to the 99th as well as by computing estimates for higher percentiles after fitting an exponential model to observed tail values. Specific details about the estimation algorithm are beyond the scope of this paper.

Data visualization

Baseline statistics can be viewed on the Baseline Statistics Visualization page. This page displays a stacked bar chart of metric percentiles for each of the hours in a typical week over the baseline period. The baseline time partitioning determines how many distinct bars are in the chart from a minimum of one (coarsest partitioning "By all hours and by all days") to a maximum of 168 (finest partitioning "By hour of day and day of week").

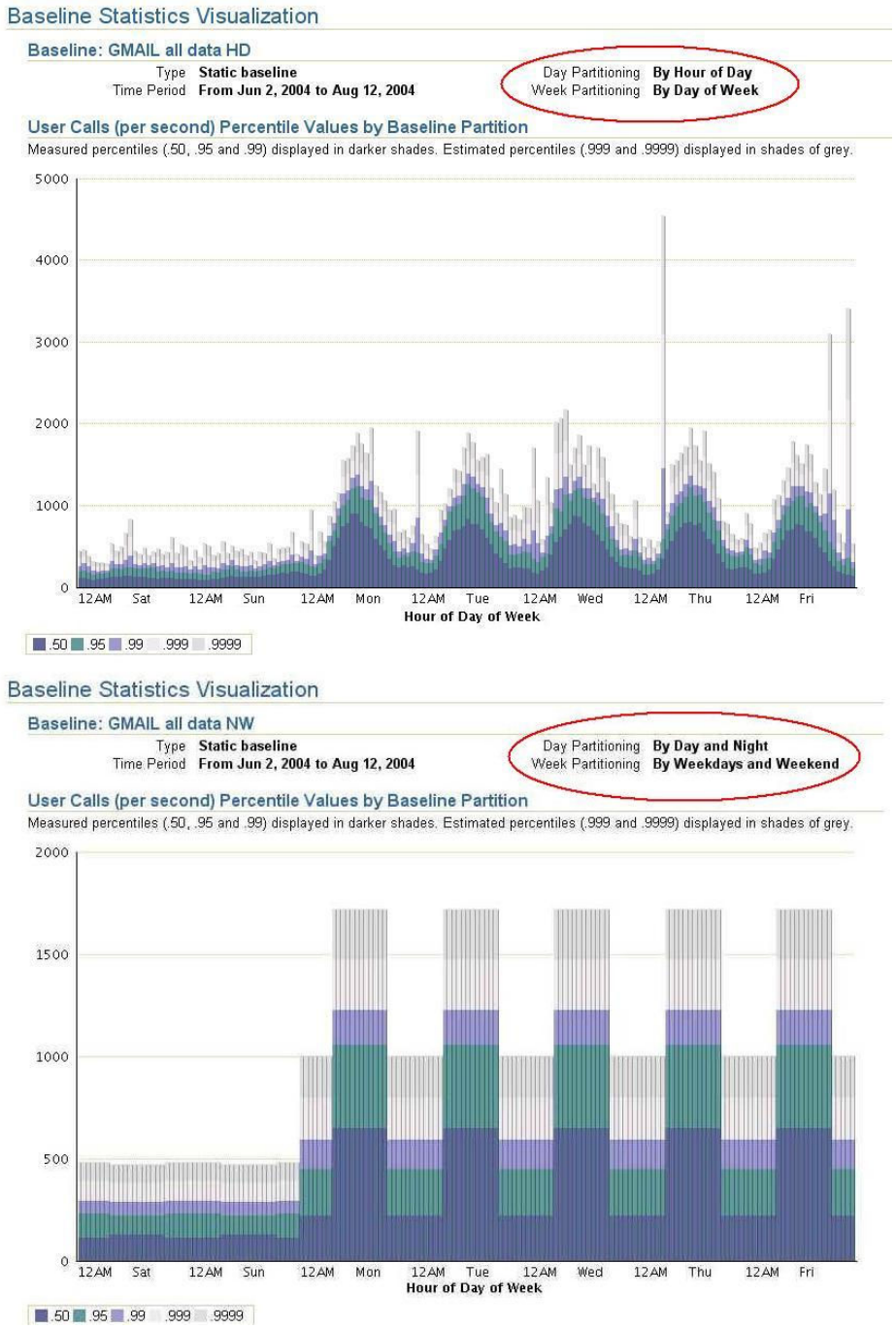

Figure 5: Data visualizations of baseline statistics for *user calls per second*

Figure 5 above shows statistics visualization pages for two baselines defined over the same time period but with different granularities of time partitioning.

In the upper chart each hour is modeled separately and we can see the daily patterning strongly evident, especially during the weekdays. However, notice also the high degree of variability in the estimated percentiles, with some spiking very high. This is because the smaller sample sizes used for each hour result in much more sensitive estimates. On the other hand, the lower graph aggregates in 12-hour chunks and produces estimates that tend to match the dominant hours within the period.

ADAPTIVE ALERT THRESHOLDS

Metric baselines can be used to establish alert thresholds that are both statistically significant and adaptive to expected variations across time partitions. Two types of statistical thresholds are supported:

- Significance level thresholds
- Percent of (trimmed) maximum thresholds

Metric alert thresholds are computed and updated at most once per hour by a database job scheduled to run at the top of each hour.

Significance level thresholds

Alert thresholds can be dynamically set by the system to values representing statistical significance as measured by the active baseline. Alerts generated by observed metric values exceeding these thresholds are assumed to be unusual events and therefore possibly indicative of, or associated with, problems (assuming problems are unusual).

The following significance level thresholds are supported:

- **HIGH**, significant at 0.95 (5 in 100) level
- **VERY HIGH**, significant at 0.99 (1 in 100) level
- **SEVERE**, significant at 0.999 (1 in 1000) level
- **EXTREME**, significant at 0.9999 (1 in 10,000) level

It is recommended that customers use significance level thresholds conservatively and experimentally at first.

Percent of maximum thresholds

The percent of maximum threshold allows the user to set metric thresholds relative to the trimmed maximum value measured over the baseline period and time partition. The measured 99^{th} percentile value is used as the trimmed maximum value. So a warning threshold level of 110% of maximum will trigger an alert when an observed metric value is 110% of the 99^{th} percentile value as measured over the baseline time partition assigned to the time of the observation.

Percent of maximum thresholds are most useful when a static baseline has captured some period of specific workload processing (e.g. Month-end) and we want to signal when the next occurrence of this processing exceeds some fraction of the baseline period value. We are not looking for statistically unusual values in this case, but rather values close to or exceeding peaks observed over the baseline period.

So for instance an online store with significant holiday peaks may capture these as static baselines for year-over-year comparisons using percent of maximum thresholds.

BASELINE-NORMALIZED VIEW

Metric value time series can be normalized against a baseline by converting each observation to some integer measure of its statistical significance relative to the baseline. EM 10gR2 includes a page to display registered metrics normalized against the current active baseline. Figure 6 below shows an example of this view for a day on which an unusual performance event was observed.

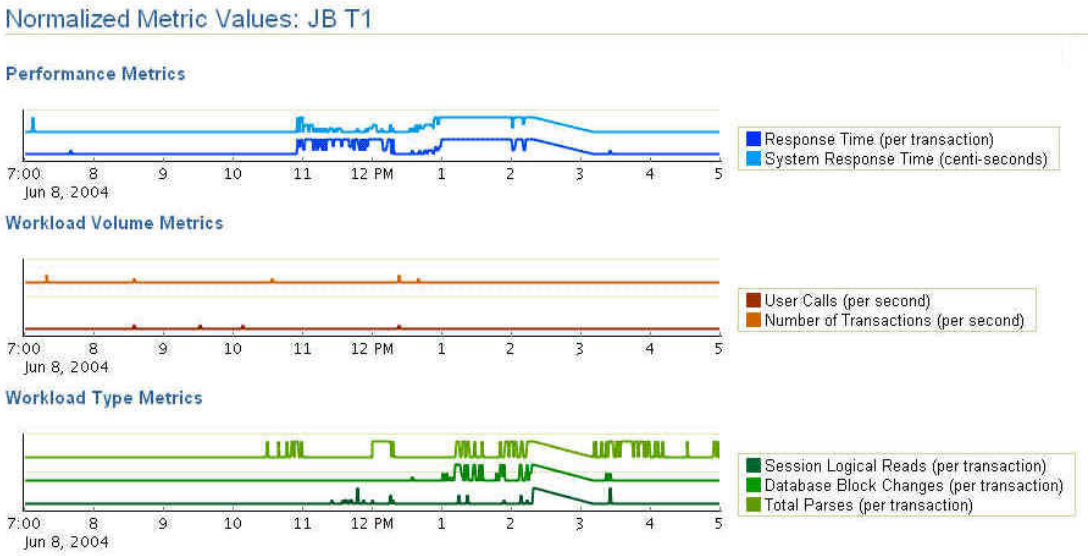

Figure 6: Baseline normalized view showing performance event and correlated workload type events

In this graph only groups of statistically significant metric values within 5-minute windows can register as events or "blips" on the chart. Metric values that are below 0.99 in significance do not register and render as "flatline". The result is a kind of seismograph of system performance. In Figure 6 we observe the time correlation of performance events with workload type events.

Several important advantages of transforming and rendering metric data in this way are discussed below.

Multiple metrics

Normalizing all metrics to statistical significance against baseline eliminates issues with axes, labels and units when trying to display multiple time series together on a page. All baseline-normalized time series share a common dimensionless Y-axis and thus can be compared to each other simply and directly.

Signal vs. noise

The normalized view suppresses all data except certain statistically significant events, which are assigned values and aggregated over 5-minute intervals in the graphs. The graphs are analogous to earthquake seismographs. Important data is filtered out, amplified and made obvious. The EM version of this graph allows for several levels of noise reduction to further isolate highly significant event periods.

Metric categories

The normalized view separates metrics into groups based on category. Metrics from the same group appear in a graph together and in related shades of color. This promotes thinking about group level events in addition to individual metric events.

For Oracle RDBMS 10gR2 the metrics are grouped as follows:

- Performance, metrics related directly to time spent in the system
- Workload Volume, metrics indicative of the quantity of workload or demand
- Workload Type, metrics indicative of the profile or shape of the workload

Event correlation and origin

All the graphs on the normalized view page share a common time interval as the X-axis. Combined with the noise-reduced display, this enables rapid visual time-correlation of events. Thus it is possible for performance events to be seen as time-synchronous with either significantly increased demand or significantly unusual workload. In other words, the high-level origins of performance problems may be quickly suggested by such visual correlations. This constitutes a first-level root cause analysis that may help administrators explain and diagnose performance events.

CONCLUSION

The metric baselines introduced in Enterprise Manager 10gR2 statistically characterize specific system metrics over time periods matched to system usage patterns. These statistical profiles of expected metric behavior are used to implement adaptive alert thresholds that can signal administrators when statistically unusual metric events occur. Assuming that systems are normally stable and performance problems rare, it is reasonable to expect that actual performance events will be highly correlated with observed unusual values in some metric or other. Thus it is hoped that baseline-driven adaptive thresholds will both reduce configuration overhead for administrators and more reliably signal real problems than fixed alert thresholds.

The baseline-normalized view visually isolates and amplifies statistically significant metric events together using common graph axes. This can provide explanatory power and deeper understanding of system dynamics.

Acknowledgements

Dr. Amir Najmi of Oracle Corporation deserves many thanks for contributing his statistical analysis and the plots in Figures 1-4. Thanks also to Jon Soule and Shivani Gupta of Oracle Corporation for work on the metric baselines project, as well as to Mark Ramacher, Graham Wood, and Gary Ngai of Oracle Corporation for their support.