Introduction

I’ve always been interested and intrigued by the processes DBAs use to monitor their databases. I’ve seen many DBAs hunched over keyboards (affectionately called “gargoyle DBAs”) continually watching sessions execute and resources being consumed. I’ve often wondered what these gargoyle DBAs are trying to catch their database system doing. Of course I know—they’re trying to catch an application or user perform a lock, generate an I/O request, or spend time in some obscure resource wait event. I think it’s time we ask ourselves why this manual, time-intensive monitoring is taking place. A more viable approach to performance monitoring is recognizing that resource consumption on databases is going to happen. The key is to know when resource consumption has happened, was it a problem, will it happen again, and at what levels.

Workload

Regardless of how many applications, users, or SQL statements we throw at them, our database systems have a physical and finite set of resources and time that can be used to service requests and provide result sets back to applications and users. The amount of work a database system produces or can produce, along with the resources consumed, in a specified time is considered its workload. In order to determine the true performance of our database systems, it is imperative that we are able to monitor the workload and relate the resources used to the various consumers against a timeline.
It was good to see that workloads are at the heart of the Foglight Performance Analysis product. The power of workload monitoring with Performance Analysis becomes apparent as you slice and dice into various workloads and the resources they consume—looking at workloads from various angles or dimensions. Performance Analysis supports a number of dimensions, including SQL statements, programs, OS users, DB users, client machines, command types, actions, modules, client info, and sessions. You can group and drill down within these dimensions to quickly find anomalies in workloads. For instance, you could easily find an application server from a server farm that is producing a higher level of transactions. Without workload monitoring, this is nearly impossible to find.

**Collecting the statistics**

Every database performance tool has to have a method of collecting statistics. In my opinion, this is where Performance Analysis shines. The collection mechanism, StealthCollect, is an agent that is deployed on the database host, attaches to the Oracle shared memory, creates a read-only shadow process of the SGA, and reads memory blocks without ever needing to log in to the Oracle instance. With a collection rate of up to 400 samples per second, the agent sends samples to a middleware machine where a set of processes takes the memory samples and stores them in a readable form—relieving the database host of any processing. This is quite different from other collection techniques that use the standard Oracle API, which relies on a SQL connection to Oracle and querying the V$ and X$ structures. Using the standard API is actually very expensive, stresses the production system, and has been known to cause the Oracle database to lock up. StealthCollect, which bypasses the Oracle API, is non-intrusive, maintains a small footprint, and offers the following benefits:

- Continually and more frequently collects samples, regardless of the current state of the database, to provide consistent and high-quality data. Other tools often fail or time out when trying to connect to an Oracle instance having performance problems.
- Offloads processing and does not stress the database system. Performance Analysis collects statistics outside the database and immediately sends the data to a middleware server where all storage, processing, analysis, and reporting are performed.
- Consumes minimal CPU and is self-adjusting. StealthCollect runs outside the database, doesn’t use any of the database’s CPU resources, and is not affected by a poorly tuned database. Further, you can specify the maximum amount of system CPU StealthCollect uses, and it also adjust its collection rate to match the amount of activity on the database.
Product Review: Quest Software’s Foglight Performance Analysis for Oracle

About the only downside of StealthCollect is that it, like any agent, must be installed on each monitored host. This can be a logistical barrier to overcome—obtaining install permissions that a DBA typically doesn’t have because of security separation within an organization. The good news is that StealthCollect should only have to be deployed or upgraded on the database host once for any version of Oracle. Quest has built the agent to dynamically support new patches so if an OPatch comes out, StealthCollect automatically calibrates for that patch.

Working with Performance Analysis

After StealthCollect collects statistics, the middleware is able, by default only, to store 90 days of performance data. With an optional repository, history can be increased to 5 years, also a suggested default. Clearly the purpose of Performance Analysis is to offer the customer a mechanism for the collection and long-term storage of statistical data to enable all types of diagnostics against this data such as trending, root cause analysis, and capacity planning.

In terms of usability, Performance Analysis is very intuitive. Much like file navigation, users expand, collapse, and drill down within dimensions, those high-level workload groupings, and quickly see representations of resources consumed for the workload. It is very easy to see patterns and identify anomalies in resource usage—quickly pointing the DBA to areas to investigate further, either with drilldowns or any number of reports. As far as screen layout is concerned, as the user navigates, they are presented with three panels: one for total resources consumed, another with a detailed listing of performance data, and one with a graph that shows the percentage of resources consumed by the detail selected in the detailed listing. This makes it easy to not only see total resources consumed for the selected workload, but also to see how that workload is affected by individual database activity.

Performance Analysis goes beyond the simple collection and presentation of raw performance data. Its strength is in its ability to aggregate raw data and offer current and historical workload analysis to quickly detect anomalies from trends and pinpoint root causes within those anomalies. The DBA is able to pinpoint true performance issues causing resource consumption issues and then validate tuning efforts. Simply put, Performance Analysis allows for the quick digestion of performance information, directs DBAs in tuning efforts, and provides valuable current and historical workload trend analysis to understand resource consumption—going far beyond just raw data or traditional performance metrics.
Compare

Change is the only thing that is constant and without change there would be no problems. At times I think I’ve built my entire database career around the concept of change. When I get called into an organization and they ask me to fix their performance problems, the first question I ask is, “What’s changed?” This, 100% of the time, relates directly to the problem. This is why I like Performance Analysis so much. It is able to compare just about anything—workload, resource consumption, performance metrics or consumers, within any time frame, and compare it against any other time frame or against a baseline. You can quickly tell the difference between what was experienced and what is considered normal. Think of the power when someone comes and shows concern that processing seemed a bit slow between the hours of 10pm and 3am and you are able to quickly compare that time period against past workloads—slicing and dicing across the instance, programs, users, machines, etc.

So instead of querying V$ tables, searching through logs, or trying to recreate the problem and analyze TRACE files, DBAs can quickly look at the time frame for the complaint, graphically see where resource consumption is out of balance, or begin comparing across normal processing periods to quickly extract the differences.

Rest assured the comparison is much more than a simple diff of SQL statements or explain plans. When the comparison is done, not only is the difference between the selected entities displayed, but detailed activity for individual components is textually and graphically shown—quickly pointing to the resources most consumed and of greatest concern for the workload in question. I have not seen such a clean and quick method for getting to root causes and identifying real tuning opportunities.
Alerts

Every database monitoring solution should have alerts that are triggered when something goes wrong. Performance Analysis is no different except the alerts are triggered by real baseline deviations or performance advisories. This is much different than tools that only alert you to a threshold. Thresholds are static and typically do not relate well to how a database is actually performing. Alerts from Performance Analysis, on the other hand, relate directly to the workload on the database and let you know when resource consumption is out of balance—quite different and very powerful to getting what I call real alerts, instead of fake alerts.

Alerts are categorized for services (database, application, hosts, etc.) and severity. A brief explanation of the alert is given, as well as important items such as change tracking events (changes in tables, indexes, execution plans, etc.) or instance components (shared pool structure, wait events, configuration, etc.) that need attention. Drilling in deeper to the alert shows detailed information about what was happening in the database, a description of the problem and an action plan that includes the key metrics, the rationale as to why the alert was raised, recommended actions, and links to additional information and other products to aid in problem resolution, such as Quest SQL Optimizer or Space Manager—everything you need to understand the alert and solve it quickly. Plus a few niceties to manage alerts like acknowledging the alert so other Performance Analysis users know someone is aware of the alert and attaching statuses and notes for future reference—making this a growing knowledge base for when other DBAs must work on the same problem in the future. Or if you’re confident in the plan of action or want to forward the alert to a different notification system, you can script actions for alerts using the rules engine.

Working with RAC

Oracle real application clusters (RAC) provides a database solution where a single database is shared across a cluster of servers—providing databases with scalability and high availability features. With RAC we are now asking our DBAs to monitor and tune multiple instances (nodes) simultaneously and with regard to each node in a RAC environment. In Performance Analysis, RAC nodes are exposed as single logical entities—making it very easy to see how each node affects the cluster. When looking at a cluster, you can easily see each node’s workload and how it contributes to the entire workload of the cluster—validating if the cluster is out of balance.
You can then drill down into each node to see how individual SQL statements, programs, users, etc. are performing on it. And in my opinion, the best thing here is that you can also do a comparison of activity between nodes and see if there are any drastic differences in node resource consumption. In just a few clicks, you can graphically see where workloads differ across the nodes along with various metrics to see where changes happened that caused the difference in workload.

Performance Analysis in a RAC environment is really not much different than for a single instance. The only real difference is the extra layer of nodes under a cluster. The interrogation, analysis, and alerts all behave the same: zero in on a workload, determine if the workload has changed, and then drill down or alert on that deviation from the norm.

**What about Oracle’s Enterprise Manager (OEM)**

Many performance tools, including Oracle’s Enterprise Manager and Diagnostic Pack (OEM/Diag), rely on Oracle’s standard API (X$ and V$ views) and built-in instrumentation. Because of this, OEM/Diag is highly coupled with the database, performs statistical collection inside (while connected to) the database, and has packages, procedures, and code on the database server that are required for much of the statistical collection and massaging of performance data. Under these conditions OEM/Diag can become just as much a performance problem as the one we’re trying to solve. And since OEM/Diag relies on performance data that is in the database itself, that data may be inaccessible if the database is overloaded or down. So when you’re having a problem, you can’t get to the data to troubleshoot the problem—sort of a “catch-22” and arguably a fatal flaw in Oracle’s architecture. To help alleviate the potential performance impact, OEM/Diag limits the workload collection mechanism to such a low frequency that the quality of the statistics becomes questionable. Couple this with the fact that OEM/Diag has limited buffer/storage space and will often age out statistical information, and a DBA will end up missing statistical data and get no real sense of any trends that are emerging within the database.

**Performance Analysis vs. Spotlight**

In addition to Performance Analysis, Quest also offers Spotlight. What’s the difference and why are two tools needed? The primary difference is that Performance Analysis focuses on the database workload while Spotlight focuses on the database infrastructure. In other words, the two tools target different layers of the database hierarchy. That hierarchy starts with the database infrastructure, which services sessions, which create workload. Spotlight focuses on the database infrastructure,
Product Review: Quest Software’s Foglight Performance Analysis for Oracle

which has traditionally been the target of database monitoring. Performance Analysis extends that traditional approach by capturing and analyzing workload.

Most people would probably agree that infrastructure monitoring is essential for any database—we simply wouldn’t put a database into production without this minimum level of monitoring. But what about workload monitoring? I think this additional level of monitoring is needed for mission-critical databases where outages or slowdowns can have a serious impact on the organization. Transaction-intensive databases are also candidates since once you’ve tuned the obviously bad SQL, you need a more comprehensive view of activity like Performance Analysis provides to troubleshoot problems and maintain performance, especially as users and data volumes grow. And you could easily argue that the tool should be deployed on all databases as a standard and because it makes DBAs more productive.

**Conclusion**

Resource consumption isn’t bad; resource consumption only causes a problem when it is above normal and reaches the limits of the system. Performance Analysis provides more than sufficient entry points to investigate the workload for databases. With the aid of comparing workloads, any DBA can easily pinpoint current or potential problems in an Oracle database and quickly generate a viable action plan to remedy the issues.

It is worth noting that Performance Analysis has been around since 1999 and is a very mature and feature-rich product. There was in no way enough time to touch on and review all the features of this product. For example, IntelliProfile, ChangeTracker, reporting, configuration, and the data warehouse are all worth noting and exploring in further detail.