solarwinds

RESPONSE TIME ANALYSIS

A Pragmatic Approach for Tuning and Optimizing Oracle Database Performance

SOLARWINDS
WHITEPAPER

INTRODUCTION

For database administrators the most essential performance question is: how well is my database running?

Traditionally, the answer has come from analysis of system counters and overall server health metrics. You might, for example, look at how many physical reads there have been within a specific period of time, the number of writers or lock statistics. Or you might look at dashboards of information about the CPU, disk I/O, memory, storage and the network.

Yet, because the primary purpose of a database is to provide end users with a service, none of these counters or metrics provides a relevant and actionable picture of performance. That is why, in recent years, an increasing number of performance experts have recommended a shift from measuring resources consumed to measuring application performance. To accurately assess database performance from the perspective of service provided, the question must become: how much time do end users wait on database response? To answer this question, you need a way to assess what's happening inside the database that can be related to end users.

Yet, because the primary purpose of a database is to provide end users with a service, none of these counters or metrics provides a relevant and actionable picture of performance. That is why, in recent years, an increasing number of performance experts have recommended a shift from measuring resources consumed to measuring application performance. To accurately assess database performance from the perspective of service provided, the question must become: how much time do end users wait on database response? To answer this question, you need a way to assess what's happening inside the database that can be related to end users.

Response time analysis (RTA) is an approach that enables DBAs to measure the time spent on query execution and, therefore, measure impact to end users. Introduced in 1999 as a database performance perspective by Cary Millsapi[1], and articulated again in 2001 by Craig Shallahamer[2,] RTA is a proven, comprehensive way to monitor, manage and tune database performance that is based on the optimal outcome for the end user, whether that user is an application owner, batch processes or end-user consumer.

RTA does for DBAs what application performance management (APM) does for IT—identify and measure an end-to-end process, starting with a query request from the end user and ending with a query response to the end user, including the time spent at each discrete step in between. Using RTA, you can identify bottlenecks, pinpoint their root causes and prioritize your actions based on the impact to end users. In short, RTA helps you provide better service, resolve trouble tickets faster and free up time to focus your efforts where they have the most impact.

This paper describes RTA in detail, and provides an overview of the tools available to perform RTA so that you can monitor and manage database performance by focusing on its impact to end users.

WHY AREN'T COUNTERS AND SYSTEM HEALTH METRICS ENOUGH?

When a database isn't performing well, or when web pages are loading too slowly, or price lookups are so slow that customers abandon the page, or batch processes don't complete in time, often the first response is to look at counters and system health metrics to find the answer. Unfortunately, this information is incomplete, sometimes misleading, and rarely easy to correlate to end user experience.

On the counter side, you might look at how many physical reads there have been within a specific period of time, the number of writes or lock statistics. This view of performance is, however, limited. Let's say you've been told an application is running slow. Upon a closer look, you see that the cache hit ratio is only 60%. What does this tell you about the root cause? Is the buffer cache too small? Is there a large full table scan occurring or some other inefficient SQL statement executing? If so, which ones? What impact does this have to the end user? None of these questions is answered by looking at cache hit ratio by itself.

On the system health metrics side, the amount of actionable information is equally lacking. Let's say CPU utilization shoots up to 90% on one of your servers, and then stays at that level for a day. Is it something that happened in the server? Is it something that happened in the SQL? If all you look at is that spike in CPU utilization, you might conclude that demand has exceeded server capacity and add a new server. What if that doesn't change the CPU utilization? Did it impact the end user in any way? It wouldn't be the first time that expensive hardware additions didn't solve the problem. You simply can't answer these questions armed with just server health information.

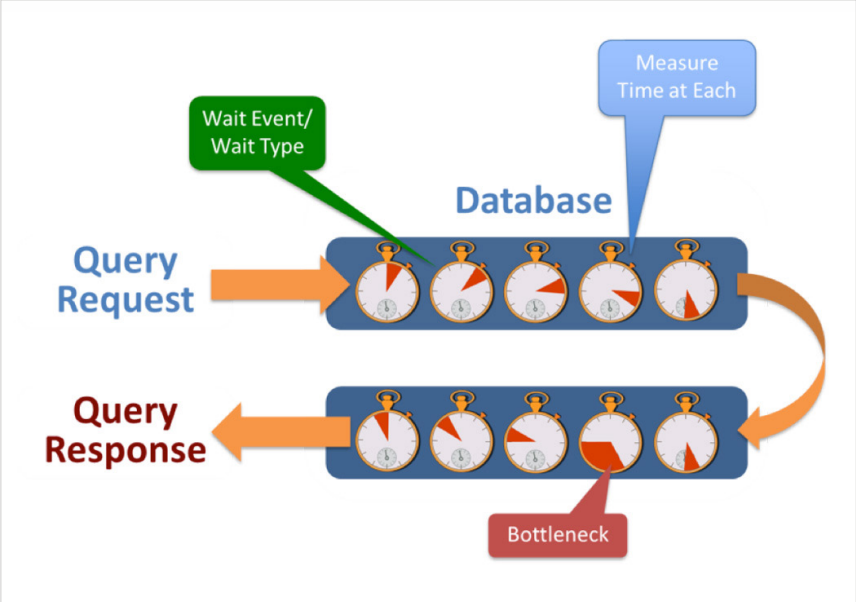
A database performance approach focused on measuring impact to end users clearly requires different information. It requires information that exposes what is happening inside the database and quantifies the impact to users.

RESPONSE TIME ANALYSIS IS ALL ABOUT WAITS, QUEUES AND TIME

At the core of RTA is the wait event or wait type. When a server process or thread has to wait for an event to complete or resource to become available before being able to continue processing the SQL query, it is called a wait event. For example: moving data to a buffer, writing to disk, waiting on a lock, or writing a log file. Typically, hundreds of wait events must be passed between a query request and response.

If an SQL query is waiting on a specific wait event more than is usual, how can you find out? How do you know what is 'normal'? How can you find out why it's waiting? And how do you fix it? That's where RTA comes in.

RTA measures time for every wait event for each query, and provides a way to look at this data in context with other queries, and in context of time, so that you can prioritize efforts that have the most impact on the end user.



In order for RTA to work, it must do the following:

1. **Identify the specific wait event that causes a delay**

   You must measure every wait event individually to be able to isolate potential problems and bottlenecks.

2. **Identify the specific query that got delayed.**

   If you know only that the whole instance is delayed, you will not be able to isolate the specific query that is causing a delay or bottleneck.

   It's important to know that some performance monitoring tools and dashboards present an averaged assessment across multiple SQL statements. If you are looking at averaged data, how can you tell which specific SQL statements are slowing performance? In order to have an RTA approach that works, you must collect performance statistics at the individual SQL statement level.

3. **Measure the time impact of the delay.**

   You must measure the wait event time—the time to complete the task and pass the work on to the next process, for each wait event. Without this information, you will not be able to completely understand what is causing the bottleneck.

Armed with all this data, you can account for each SQL and each resource utilized by an SQL. And then you can accurately pinpoint the root cause of delays, inside or outside the database.

WHERE DOES THE DATA REQUIRED FOR RESPONSE TIME ANALYSIS COME FROM?

There are a number of ways to obtain the performance data required for RTA, including Oracle Enterprise Manager (OEM, sometimes also referred to as Grid Control), Trace files, and Oracle's V$SESSION tables. Each has strengths and also important limitations. For example, most are time- intensive, limited to point-in-time views, and accessible only to highly experienced DBAs. Additionally, continuous performance monitoring tools complement existing database management tools and can greatly simplify and speed RTA.

ORACLE OEM WITH AWR AND PERFORMANCE/TUNING PACKS: PROS AND CONS

As an Oracle DBA, the most obvious tool for RTA-based performance management might be Oracle OEM with Automatic Workload Repository (AWR) and Performance/Tuning Packs. Expert OEM users can find the three key elements of RTA—query, wait and time—but it can take considerable time, expertise and detailed analysis. Developers or managers, for example, likely won't be able to successfully use OEM, and will rely on the DBAs to access and interpret the data for them.

Limiting its usefulness even further is the fact that OEM's local collection of performance data places a load on production servers and requires users to have access to the production database. This makes OEM use very restrictive, and not desirable for real-time RTA.

Furthermore, OEM isn't even available for Oracle Standard Edition (SE) implementations.

Continuous monitoring, via trace

Trace produces rich precise information that might not be available in any other way, and does capture all the SQL statements.

However, the drawbacks of trace are many. To begin with, tracing is turned on selectively, and it produces no historical/trending information. You must know in advance when and where a problem is going to occur, turn on tracing for that session, and watch to see what happened. This means that trace isn't useful for 24/7 monitoring, for complete performance analysis of production systems, or for proactive performance management.

Trace also produces high overhead. If you're using it for problem-solving, it will add to overhead at the worst possible time. In fact, the load that tools like trace put on a database server are one of the reasons some DBAs don't want to do RTA; they mistakenly believe that trace is the only way to get the information needed to do effective RTA.

Another sometimes overlooked problem with trace is that it produces almost too much information. You need a skilled DBA and lots of time to accurately sort through it and assess its meaning. This makes it especially inappropriate for developers and others who aren't highly experienced DBAs.

Finally, trace can skew your assessment of root cause of performance problems. For example, consider a situation in which you see 95 of 100 sessions are running well, but five sessions have spent 99% of their time waiting for locked rows. If you trace one of the 95 "good" sessions, you may think you don't have any locking issues, and you will likely spend time trying to tune other SQL queries that may not be so critical. Yet, if you happen to trace one of the five "bad" sessions, you might think you could fix the locking problem and so reduce the wait time by 99%. Neither assessment is entirely accurate, and any effort you make based on that assessment will fail to address the real cause of the issue. That can quickly lead to a great deal of wasted time and resources.

Oracle's response time tables: V$SESSION views

Oracle's V$SESSION views also provide wait event information. Some of these tables are point of time views, while others are historical. The more useful of these views are described below.

As with OEM and trace, however, there are limitations. None of these views provides continuous information. None provides a consolidated view of performance data. None are easy to use to achieve actionable results.

VIEW	DESCRIPTION	EXAMPLE
V$SESSION	Use V$SESSION to find out what sessions are active, what SQLs they are running (SQL_ID), whether that SQL is "waiting" (actively waiting for a resource), or "waited" (it did wait but is active now somewhere else, reading data from memory or actually processing on CPU)	V$SESSION SID USERNAME SQL_ID PROGRAM MODULE ACTION PLAN_HASH_VALUE * ROW_WAIT_OBJ#
V$SESSION_ WAIT	Note that V$SESSION_WAIT was added to V$SESSION in Oracle 10g and higher. This view includes all the wait events, and shows what the session or query is currently waiting on.	V$SESSION WAIT SID EVENT P1,P1RAW,P2,P2RAW,P3,P3RAW STATE('WATING','WAITED...') Oracle 10g added this info to V$SESSION
V$SQL	Using SQL_ID from V$SQL to understand the plan used by Oracle.	V$SQL SQL_ID SQL_FULLTEXT PLAN_HASH_VALUE EXECUTIONS DISK_READS BUFFER_GETS
V$SQL_PLAN	Use PLAN_HASH_VALUE from V$SQL to understand the plan used by ORACLE.	V$SQL SQL_ID PLAN_HASH_VALUE
DBA_OBJECTS	Join with ROW_WAIT_OBJ# from V$SESSION. This view tells you what specific objects or tables the SQL us accessing.	DBA_OBJECTS OBJECT_ID OBJECT_NAME OBJECT_TYPE

CONTINUOUS PERFORMANCE MONITORING TOOLS

Continuous performance monitoring tools mark the next evolution of tools available for DBAs implementing RTA. These third-party tools, which typically complement existing database management tools like OEM, aim to simplify performance management by consolidating performance information in one place.

However, even these tools may have limitations, and you should carefully evaluate a tool before making an investment.

The best continuous performance monitoring tools for RTA will meet the following criteria:

- Identifies and measures the three key elements needed for RTA—if the tool doesn't provide all of these, it just won't work:
    - Identifies the specific SQL query that got delayed
    - Identifies the specific bottleneck (wait event) that causes a delay
    - Shows the time impact of the identified bottleneck

    It's important to pay particular attention to how the tool collects data, and where it collects data from.
- Provides continuous 24x7 monitoring with access to historical trend data
- Puts a negligible load on the system, less than %1 (some of these tools rely on trace to gather performance data, which can create a load on the monitored server)
- Can be used by non-DBAs, like developers and managers, to assess performance (so DBAs can spend less time explaining and more time doing). Be sure to ask these questions:
    - Is it easy enough to use and understand that a non-DBA can use it?
    - Does it require a user to have access to your production database?

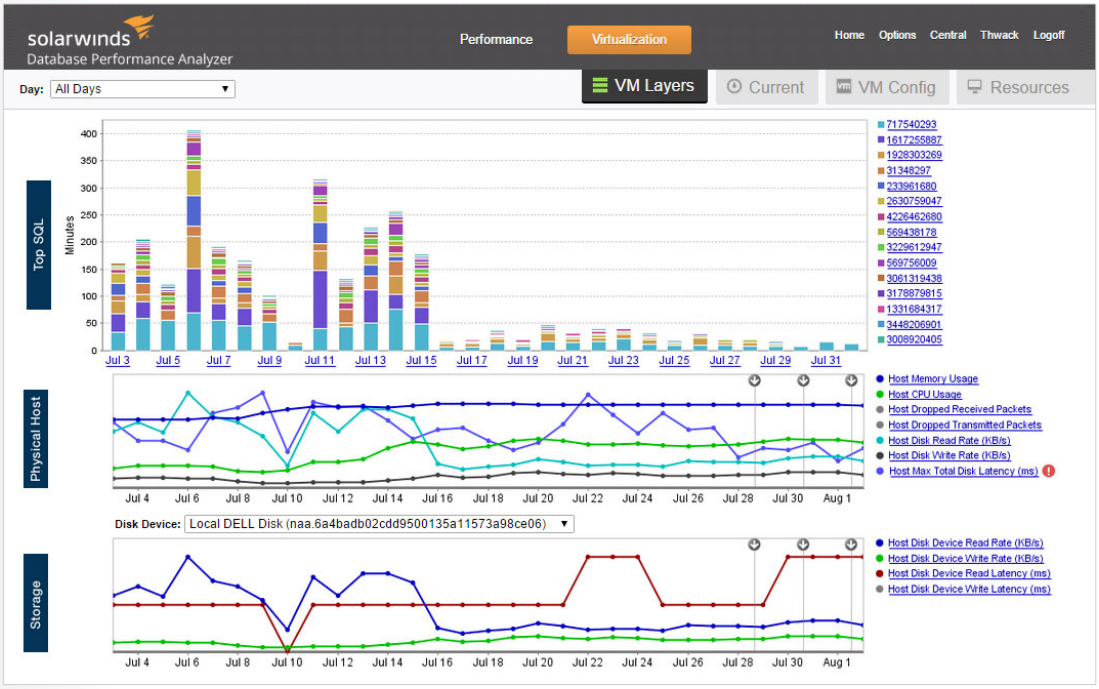
Unless the tool meets all these requirements, it will be of limited use in performing RTA.

IT'S TIME FOR RESPONSE TIME ANALYSIS

RTA provides a systematic way to discover and measure performance as it relates to user experience. RTA measures a query end to end, from request to response, all the waits encountered in between, and all the resources used by a query during processing. RTA puts this information into context with other queries, and provides historical trend information to make it possible to identify and fix the bottlenecks with the greatest impact to end users. RTA can transform database tuning from a reactive process done in crisis mode late at night or on the weekend to a proactive practice tied to the end users it serves.

HOW CAN DATABASE PERFORMANCE ANALYZER HELP?

Database Performance Analyzer (DPA) from SolarWinds (NYSE: SWI) provides the fastest way to identify and resolve database performance issues. DPA is part of the SolarWinds family of powerful and affordable IT solutions that eliminate the complexity in IT management software. DPA's unique Multi-dimensional Database Performance Analysis enables you to quickly get to the root of database problems that impact application performance with continuous monitoring of SQL Server, Oracle, SAP ASE and DB2 databases on physical, Cloud-based and VMware servers.



For additional information, please contact SolarWinds at 866.530.8100 or e-mail **sales@solarwinds.com.**

LEARN MORE

DOWNLOAD FREE TRIAL

Fully Functional For 14 Days

[1] "**Performance Management Myths and Facts**," by Cary Millsap. Copy available courtesy of Oracle Corp. at: http://method-r.com/downloads/cat_view/38-papers.

[2] "**Response Time Analysis for Oracle Based Systems**," by Craig Shallahamer. http://resources.orapub.com/product_p/rta.htm