CONFIO is now part of solarwinds

Response Time Analysis

A Pragmatic Approach for Tuning and Optimizing SQL Server Performance

By Dean Richards

Confio Software
4772 Walnut Street, Suite 100
Boulder, CO 80301

www.confio.com

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 instance performance from the perspective of service provided, the question must become: how much time do end users wait on a response? To answer this question, you need a way to assess what's happening inside the database instance 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. Sometimes referred to as waits-and-queues performance tuning, and detailed in the paper "SQL Server 2005 Waits and Queues" by SQL Server expert Tom Davidson of Microsoft [i], 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 instance performance by focusing on its impact to end users.

Why aren't counters and system health metrics enough?

When a database instance 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 query 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 query? 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 instance 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 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 a query waits on a specific wait event more than 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.

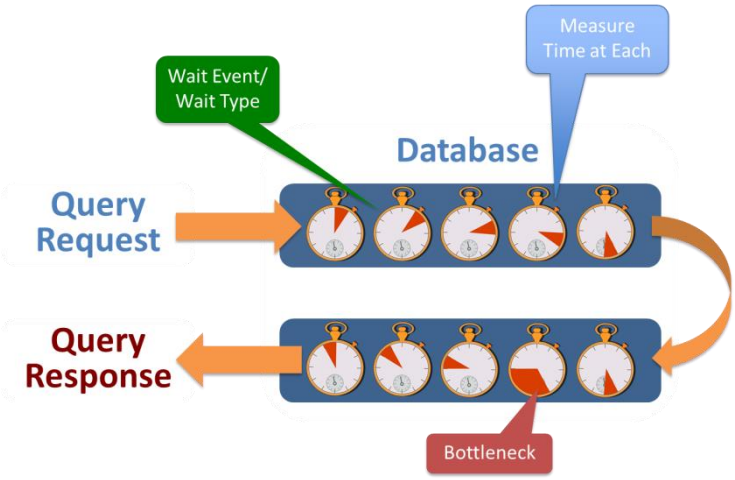
In order for RTA to work, it must do the following:

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 queries. If you are looking at averaged data, how can you tell which specific queries are slowing performance? In order to have an RTA approach that works, you must collect performance statistics at the individual query 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 query and each resource utilized by a query. And then you can accurately pinpoint the root cause of delays, inside or outside the database instance.

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 existing database management tools and Trace files. 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.

Continuous monitoring, via trace

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

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 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.

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, 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 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
- Doesn't put a load on the system (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 instance?

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 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.

About Confio Software

Confio Software, now a part of the SolarWinds family, builds award-winning database performance analysis tools for DBAs and developers. SolarWinds Database Performance Analyzer (formerly Confio Ignite) improves the productivity and efficiency of IT organizations. By resolving problems faster, speeding development cycles, and squeezing more performance out of expensive database systems, Database Performance Analyzer makes DBA and development teams more productive and valuable to the organization. Customers worldwide use our products to improve database performance on Oracle, SQL Server, Sybase and DB2 on physical and virtual machines.

For more information, please visit: http://www.confio.com/performance/sql-server/ignite/

[i] http://msdn.microsoft.com/en-us/library/cc966413.aspx