Scientifically Evaluating the Anticipated Effect of Alternative Performance Solutions

Latest version always available at http://www.orapub.com

This page has been intentionally left blank.

Table of Contents

Scientifically Evaluating the Anticipated Effect of Alternative Performance Solutions

Craig A. Shallahamer (craig@orapub.com)
Original: October 17, 2009

Version 2e, November 6, 2009

Abstract

It is possible to scientifically evaluate the anticipated effect of alternative performance solutions by creatively integrating a quantified response time based performance analysis, application workload metrics, and standard performance mathematics. Historically, the performance analyst arrives at one or two solutions and based upon their experience and the associated implementation pain decides the best course of action. Because this method is highly experience based and systems are becoming increasingly complex, in most cases *why* the solution will improve the situation is not quantified and rarely is there any anticipatory analysis performed about *how* the proposed solution will affect the performance situation. This paper introduces a methodology that quantifies anticipated solution benefits, which dramatically reduces the risk of unanticipated solution effects, enables objective evaluation, and saves everyone time.

Introduction

Before I can dive into the actual methodology, a few introductory and supporting topics need to be introduced. I will start with providing an understanding as to where this method fits into the entire performance optimization process. Then I will draw a distinct line between diagnosis and analysis. This distinction is important because it will more clearly show when this method is best applied during the entire performance optimization process. Next, I will focus on contrasting the various performance optimization philosophies and where this method naturally fits and doesn't fit. Finally, I will introduce, with figures, how this method works including a slightly more detailed step-by-step process. I will close with pointers about where you can learn more on how to scientifically evaluate alternative performance solutions. Let's get started!

Defining Performance Analysis

Performance analysis is distinct from many aspects of the entire process of improving performance. I commonly look at the entire process in seven steps. Each is quickly presented below.

1. **Define Problem.** Before any work begins there must be a clear understanding of the problem. While seemingly obvious, it is very common for work to begin with a vague or

even incorrect problem definition. If you cannot "check the box" or proclaim, "I solved the problem" when your work is complete, then the clear problem definition is not actually clear.

2. **Gather Data.** Unless you're guessing, you'll need actual data about what is/was happening during the problem time. Common Oracle focused data sources are `v$sysstat` and `v$time_sys_model`. Oracle is increasingly gathering operating system activity and making it available to the performance analyst via views such as `v$osstat`. Even better, Oracle's Statspack and AWR facilities are both gathering and storing more and more data.
3. **Diagnose Problem.** Before solutions can be discovered, an understanding of the problem situation must be gained. This is the process of diagnosis and it absolutely critical. By using an Oracle response time analysis (ORTA) based approach, this process becomes quantified, methodical, and fast. The diagnosis time is shortened so you can focus on the more analytical aspects of performance optimization.
4. **Discover Solutions.** Once the problem has been diagnosed, a number of possible solutions will become apparent and should be listed. When using OraPub's 3-Circle analysis method combined with an ORTA, a number of possible solutions will always result.
5. **Analyze Solutions.** Once solutions have been discovered, they need to be ranked for possible implementation. Because the solutions have not yet been implemented in the production system, their intended effect is truly anticipated. This is typically where intense discussion occurs and much of this is based on experience, folklore, politics, and various other situational winning tactics. This is no way to rank solutions and ultimately choose which is to be first implemented. What is needed is a scientific and objective method to rank solutions so the discussion can be focused on important issues such as anticipated performance improvement, application disruption, budget, and time to implement. The method described in this paper provides this objective and scientific ranking capability.
6. **Implement Solutions.** When solutions are analyzed, one or perhaps two solutions are chosen for implementation. Carefully plan their implementation and then do it.
7. **Verify Solutions.** Once the solution(s) is implemented, performance must once again be monitored to ensure the anticipated effect is occurring. Performance optimization is a cyclical effort and every change is likely to dramatically shift the performance situation providing another round of pure optimization enjoyment. If performance is not meeting requirements and there is time and budget to continue, then proceed back to the Problem Definition step to begin once again.

Diagnosis Versus Analysis

Database administrators commonly ask me why I separate diagnosis from analysis. I explain that diagnosis is the process of understanding and developing a clear picture of the situation resulting in knowing where the problems reside and finding reasonable solutions that directly remedy the problems. Analysis can then begin with a focus on determining which solution should be first implemented. That is, each solution is objectively evaluated, combined with business objectives and the realities of production systems and then ranked. In contrast, today most performance work is highly experienced based, and therefore, one's perspective and the associated implementation pain determines the best solutions.

Here are two short of examples of where diagnosis has taken precedence over analysis. If Angie is a fantastic SQL tuner, she is very likely to look at the problem as a SQL statement issue, and therefore, a SQL tuning solution. Whereas, an Oracle "guru" will most likely find fault/opportunity in the Oracle configuration and recommend a tweak to Oracle's instance parameters. To combat this natural diagnosis bias we all have, I developed the OraPub 3-Circle analysis method. The OraPub 3-Circle method helps (some say focus) DBAs to derive a number of solutions regardless of their inherent bias. But still, this does not address scientifically ranking the solutions.

A true analysis goes beyond diagnosis and solution discovery as it provides both a quantitative and objective method to determine which of the solutions will best serve the organization. The analysis will combine both technical and non-technical aspects, quantitatively represent each solution, and provide ways to objectively rank the solutions.

For example, the analysis method outlined in this paper, presented in the last chapter in my book *Oracle Performance Firefighting*, and taught in OraPub's *Advanced Oracle Performance Analysis* course enables the anticipated solution benefits to be quantified and indicates how they change the performance situation. Another important factor is presenting the analysis results in such a way that both technical and non-technical people can understand the situation. This creates an atmosphere of dialog and cooperation where the possible solutions can be scientifically, objectively, and calmly discussed.

To summarize, diagnosis exposes and brings solutions to our attention, whereas analysis enables a scientific understanding of the expected benefits so a more mature discussion, solution selection, and implementation is possible.

Contrasting Analysis Philosophies

To recognize the value of an Oracle response time analysis, it is helpful to understand how it compares and evolved from other performance ways of thinking. As I spend more and more time focused on Oracle performance optimization, the more I see that people's view of performance optimization is actually quite philosophical. In fact, I'll go so far as to say their performance optimization worldview affects the very core of their performance related work.

Let me give you an example. If my performance philosophy is centered on performance ratios, the way I look, think, talk about, explain to others, and relate to Oracle systems will be using ratios. In contrast, if my performance philosophy is response time based, I will immediately look at any performance situation in terms of service time, queue time, workloads, and arrival rates. It's just the way one looks at the Oracle performance world, and it also explains why it can be so painful to switch from one performance worldview to another.

There are a few core Oracle performance philosophies: ratio analysis, wait event analysis, session profiling, and response time analysis. I will quickly comment on each.

Ratio Analysis was birthed when Oracle performance analysis started. It works somewhat well, but because Oracle provides more detailed performance metrics then it did when Oracle was first released, ratio analysis is no longer the quickest or most reliable diagnosis method. A ratio analysis worldview practitioner will do anything to bring a ratio to its optimized value and then declare success.

Wait Event Analysis was birthed when Oracle instrumented its kernel with the release of Oracle version 7. Essentially, Oracle developers inserted additional kernel code, which provides timing information for optimization purposes. As DBAs, we can use this information for our performance optimization work. A wait event analysis worldview practitioner will determine the top wait event and do whatever it takes to move it down the top wait event list, and when this occurs declares success (or at least some success).

Response Time Analysis (RTA) is centered on timing and quantifying the users' experience (at least in part). Response time is defined by classic performance and operations research mathematics and also by the computing community as simply service time plus queue time. Real differences, but not true conflicts, arise in defining exactly what is service and queue time. In summary, it depends on your perspective. An IO vendor will look at service and queue time very differently, but not incorrectly, compared to a CPU chip manufacturer or a network specialist. So in reality, there are many response time component definitions. The correct posture is to ensure there is no missing time and no double counting of time. Conflicts arise when we forget to strive towards serving the user community.

Oracle response time analysis goes beyond wait event analysis because it includes not only Oracle wait time but also CPU consumption time. In addition, a true Oracle response time analysis transforms the raw response time numerics into a classic response time form: service time plus queue time equals response time. Since this approach focuses on time and includes more than Oracle wait time, it is a more inclusive method, and therefore, reduces the chances of misdiagnosis. In other words, it better reflects what a user, a session, a group of sessions, a system, an instance, etc. is experiencing. A response time worldview practitioner strives to reduce response time in hopes that time is a significant part of the user's experience.

Session Profiling focuses on what a specific user or process is doing and experiencing. While an important part of most performance diagnosis, a session profile worldview will see performance from a narrow perspective (e.g., single session) and will be at risk of missing how other activity impacts the session and visa-versa. For example, are the scattered reads the result of the session's SQL, a small buffer cache, other executing SQL, or some combination? In the extreme, the performance analysis becomes very exclusive to other activity, or perhaps better said, secluded from other activity. There is a real risk here of misdiagnosis because all computing and database systems strive to share scarce resources, so by design no one user, session, or process operates independently from another. A session profile worldview practitioner strives to reduce the session response time with limited regard to how it affects other sessions and how other sessions affect it.

Scientifically evaluating the anticipated benefits of alternative performance solutions is not a performance philosophy or worldview. However, it does naturally fit well into response time philosophies and aims to make them scientific and objective.

How This Methodology Works

In the field of Oracle performance optimization, I suspect most people would agree there is too much discussion and arguing and hand ringing and not enough quantitative analysis. Even when numerics are used, they typically do not represent the user experience and rarely attempt to quantify the solution affect. Anticipating the solution effect is deemed voodoo or the stuff

that the capacity or infrastructure group does, and besides there is too much mathematics involved. This is unfortunate, especially since we do this informally all the time and most DBAs get a rush when solving a difficult technical problem. The good news is scientifically analyzing the anticipated solution effect is not nearly as difficult as most DBAs think. In fact, it opens up a whole new world to the performance-minded DBA.

It turns out that by creatively integrating an Oracle response time analysis, the derivation of key performance parameters, and classic performance mathematics performance solution impact can be anticipated and quantified. It's not rocket science, but it is science because it is repeatable and can be tested over and over again. This method draws from the respected disciplines of Oracle response time analysis, queuing theory, performance modeling, predictive analysis, and by creatively integrating them.

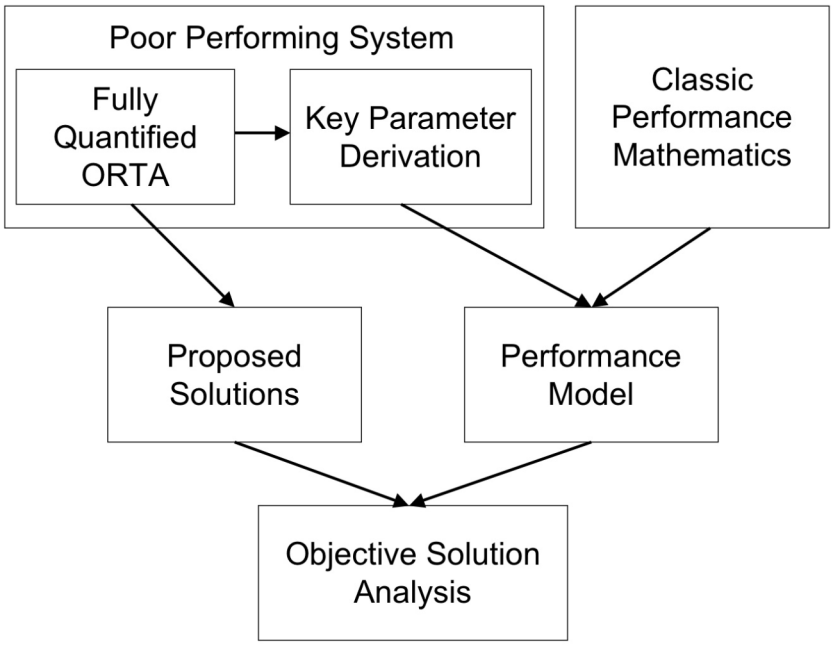

Figure 1. *Shown are the main method components, their flow, and how they relate to one another.*

The various method components are shown in Figure 1. In a nutshell, the poor performing system undergoes a quantified Oracle response time analysis and a few key parameters are derived (e.g., service time). In parallel, proposed solutions are derived as part of a standard diagnosis process, and the performance model is constructed by combining classic performance mathematics and the key parameters. Next, each proposed solution is cleverly integrated with the performance model yielding the anticipated effect. When applied to all potential solutions, each solution can be compared side-by-side both numerically and graphically. Lastly, they can be objectively discussed and when viewed in conjunction with business and application disruption concerns, the best solution rises above and its implementation planned.

While Figure 1 is helpful because it shows conceptually how the method flows, I suspect you would like more detail. This is especially true if you are not familiar with the classic performance mathematics. Please keep in mind that this is a short technical paper aimed at introducing you to the method and is not intended to be a substitute for further study and classroom instruction. As a result, I will keep this section very short and to the point. Here is a step-by-step summary of the steps involved to anticipate solution benefits.

1. **Perform an Oracle response time analysis (ORTA).** An ORTA transforms the performance situation into a response time, mathematics familiar format. The benefits are:
   - The bottleneck is easily and clearly seen bringing team consensus.
   - Solutions will be spot-on addressing the real problem resulting in quicker resolution.
   - The interaction between Oracle, the operating system, and the application workload becomes clear, which checks and strengthens our understanding of the performance situation.
   - The total CPU consumption and total non-idle wait (queue) time is quantified. These are two key parameters that are used not only to aid in problem diagnosis, but also used to mathematically represent the performance situation.
2. **Determine workload metric.** The selected workload metric must have a strong correlation with the overall system bottleneck from both an operating system and Oracle perspective. For example, if the operating system bottleneck is CPU, Oracle is consuming 85% of the server's CPU, and Oracle's top wait event is cache buffer chain latching, a good workload metric is Oracle logical IOs (i.e., buffer gets). The key is to know and understand that the workload metric is the primary stress on the overall system. It is very useful to be able to rank application SQL activity by the workload metric, since SQL tuning and workload balancing are common solutions.
3. **Determine total workload.** Based on the workload metric, determine the total workload activity over the sample interval. For example, if the workload metric is physical IO reads and the performance diagnosis is based on a one-hour Statspack report, then determine the total number of physical IO reads that occurred during this one-hour report interval.
4. **Determine total CPU consumption and non-idle wait time.** CPU consumption can be gathered from either `v$sysstat` (statistic `CPU used by this session`) or from `v$sys_time_model` (statistics `DB CPU` and `background cpu time`). The system time model data is more accuate than `v$sysstat` CPU consumption because Oracle processes, about once every six seconds, ask the operating system for their CPU consumption and make this information available via the performance views. The non-idle wait time is gathered from `v$system_event` and contains all non-idle wait time during the sample interval. Usually, simply using the wait time from the top five wait events is satisfactory.
5. **Calculate the classic performance metrics.** Once the above steps have been completed, the sample interval time, total workload, total CPU consumption, and the total non-idle wait time are combined to calculate the classic performance metrics: service time, queue time, and arrival rate. Service time is the CPU consumed to service a single workload unit, for example 1.28 ms/PIO. The queue time is the non-idle wait time associated with a single workload unit, for example 1.04 ms/PIO. The arrival rate becomes the average number of workload units that Oracle processed during the sample interval, for example 0.499 PIO/ms. How long it takes to process a single workload unit is called the response

time. The response is the sum of the service time and the queue time, for example 2.32 ms/PIO. These are classic performance parameters but have taken on a very Oracle performance specific flavor.

6. **Plot the performance situation.** While by itself, this is not all that useful, it clarifies what we're doing and sets us up for the next step. Create a graph with the response time as the vertical axis and the arrival rate as the horizontal axis, and plot the response time based on the arrival rate. Figure 2 is an example of what this looks like using the numbers in step five above.

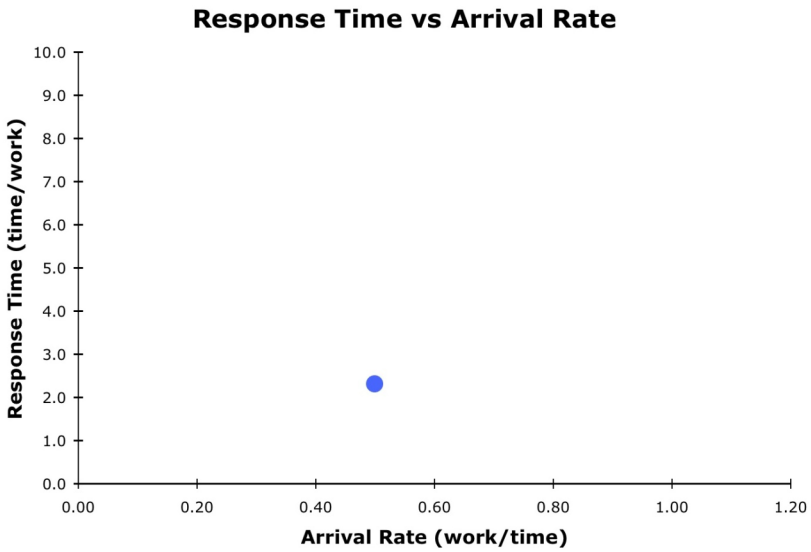

Figure 2. *Shown is a plot representing the performance situation based on a single one-hour Statspack report. To arrive at this point, the situation must be analyzed based on an ORTA and key performance metrics must be calculated.*

7. **Extend the response time plot** by combining the single data point (shown in Figure 2) with performance mathematics and altering the arrival rate in both directions. Because Oracle systems behave similarly to road traffic, telephone networks, and fast food restaurant lines (i.e., queuing systems), it is possible to take a single performance snapshot, fashion it in queuing theory terms (what we have done in the previous steps), and then draw queuing theory based conclusions.

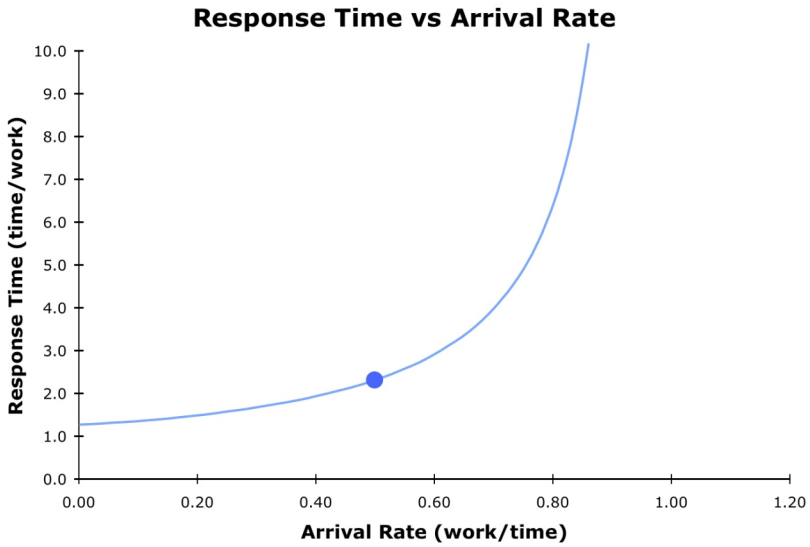

Figure 3. *Because Oracle systems behave like other queuing systems, we know that the single data point plotted in Figure 2 resides on a response time curve. By combining the performance situation with performance mathematics and altering the arrival rate (decrease and increase), the full response time curve appears.*

Figure 2 shows we can place a single point on the classic response time graph with only a single performance activity snapshot and classic performance mathematic parameters derived (arrival rate, response time). Even more important, because Oracle systems behave like other computing systems, we know that *this single point resides on a response time curve*. Figure 3 is simply the single-point queuing situation with the arrival rate decreased (line to the left of the point) and increased (line to the right of the point). Notice that Figure 3 still has the single point shown in Figure 2, but the response time curve has been extended both before and after the single point.

The Figure 3 graph represents the performance situation during the example one-hour Statspack period. This is called the baseline as all our possible performance solutions will be viewed and evaluated based on this baseline situation.

As nearly every person feels in their gut, the dot in Figure 3 represents a bad and truly unfortunate situation. Whatever the performance improving situations result from our diagnosis, they had better move the Figure 3 dot out of the steep portion of curve. This is exactly what our various tuning strategies are supposed to do!

8. **Graph each possible solution and compare to the baseline.** My teaching experiences have clearly shown it is difficult for DBAs to learn how their proposed solution affects one or possibly two of the queuing theory parameters (e.g., service time). In my Advanced Oracle Performance Analysis course, it takes an entire day of teaching before this becomes clear and usable. Let me provide two examples. First, if the Oracle instance is tuned (e.g., changing an instance parameter) then it is likely service time will decrease because each unit of work will require less CPU resources. Second, if a physical IO read intensive SQL statement is tuned the arrival rate will decrease because the SQL will generate less reads per second. This is one of the most difficult aspects to learn, but the

insight one receives is truly profound as a deep understanding of how and why solutions affect our systems becomes wonderfully clear.

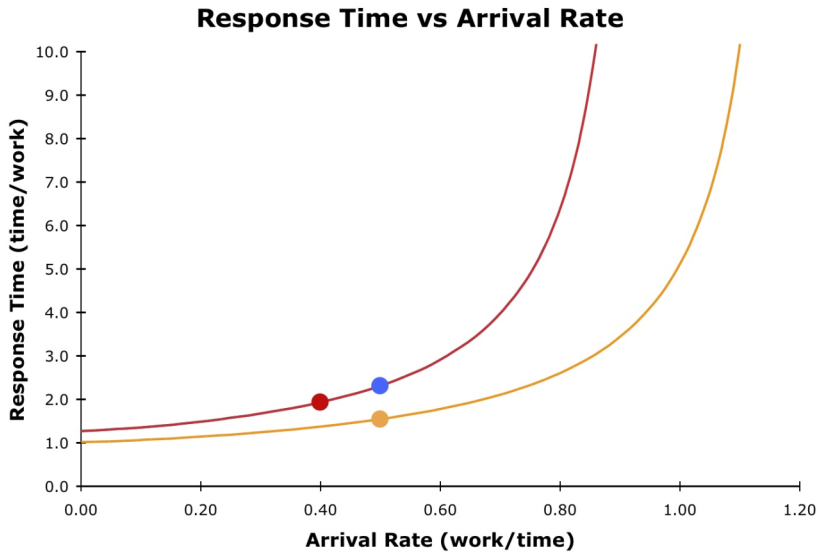

Figure 4. *With an understanding of how our proposed solutions affect the key response time parameters, we can plot the alternative solutions side-by-side. This allows everyone to objectively compare and contrast a number of solutions both numerically and graphically.*

Figure 4 shows the baseline blue plot (blue line is behind the red line and the blue dot is in the upper right) and two alternative performance solutions plotted (red is in the lower left and the orange in the lower right). The red (lower left) alternative situation is based on a net 20% decrease in physical IO reads because the key SQL statement was tuned. Because only the arrival rate decreases and the response time curve does not shift, the red dot moves to the left away from the blue dot (upper right) baseline situation. The lower right orange dot and curve is the result of implementing a faster IO subsystem. Because the IO subsystem is faster and can process more physical reads per second, the orange line drops below the baseline blue line and shifts to the right. The analysis of these alternative performance situations and understanding how the situation shifts is something that takes study, patience, and practice with real systems. This topic is part of what is taught in both my Advanced Oracle Performance Analysis and my Forecasting Oracle Performance courses.

As you can see, the above steps are applied to each of the proposed performance solutions. Then the solutions can be evaluated side-by-side or even on the same response time graph! This process allows both a numerical and visual comparison yielding a more objective discussion about which solution or solutions to implement.

This method has been tested numerous times, and it works. Don't expect a high precision forecast. In fact, I tend to use words like "anticipate" or "direction of change" rather than "prediction," "forecast," or "capacity planning." The performance model built is very simple

and is not robust enough for high precision forecasts. But, it is good enough to understand how, why, how much of a change results, and most importantly, enables a scientific, quantifiable, visual, and objective performance solution comparison!

Doing This Yourself

This paper is merely an introduction. In a way it's a tease and meant to wet your appetite so you will be motivated to try this for yourself. If you really want to learn how to effectively and responsibly use this methodology, there are two additional sources available. First, the last chapter of my book, *Oracle Performance Firefighting* introduces this technique step-by-step with an actual working example (before and after). If you really want to learn this method and also receive personal instruction, consider attending OraPub's two-day *Advanced Oracle Performance Analysis* course. This course is solidly and exclusively focused on anticipating and evaluating alternative performance solutions. To see if a class is scheduled in your part of the world, just go to www.orapub.com or email me about requesting a class in your part of the world. I also teach on-site as well, offer consulting services, as well as mentoring/coaching services. OraPub is all about Oracle performance management, and we would love to work with you in this endeavor.

Concluding Thoughts

Performance by guessing, wait event analysis, session profiling, and even response time analysis is simply not good enough. We can do better…and have fun at the same time! There is a way to perform a solid diagnosis and scientifically evaluate our proposed solutions. Imagine the reduction in meetings and arguments over the best solutions!

I hope you enjoyed reading this short paper. Oracle performance management is what I, as the founder of OraPub, do for a living. I hope this paper caught your interest, and you will be motivated to see for yourself how to integrate an Oracle response time analysis, performance metrics, and classic performance mathematics to scientifically anticipate and evaluate your proposed performance solutions!

About The Author

Craig is a researcher, writer, and teacher for 1000s of DBAs. He has published over 24 technical papers and authored the books, *Oracle Performance Firefighting* and *Forecasting Oracle Performance*. He is the creator and teacher of the popular *Oracle Performance Firefighting*, *Oracle Forecasting & Predictive Analysis*, and the new *Advanced Oracle Performance Analysis* courses. After nine years with Oracle Corporation, in which he co-founded the Core Technologies and the System Performance Groups, in 1998 he left Oracle to start OraPub, Inc. He is a passionate educator, a pinnacle of Oracle knowledge, and an engaging instructor. His entertaining presentation style combines with his depth of Oracle experiences to make each class unique. He has taught 1000s of DBAs on 6 continents in 23 countries!

Inquiries

To inquire about OraPub's training, please visit either http://training.orapub.com or http://www.orapub.com

To inquire about speaking engagements, please visit http://resources.orapub.com/RequestSpeaker.asp

To inquire about Oracle performance management consulting, on-site training, and mentoring/coaching services, please email me directly.

Craig can be emailed directly at craig at orapub.com.