Preventing the Post-Production Performance Problem

How confident are you that those new features you're adding to your production application will be fast and efficient? What if they're not? You need a process that finds inefficient code earlier, and a process to fix the inefficiencies that evade early detection. One process can accomplish both goals.

by Cary Millsap

Oracle tracing is an economical but rich performance data source.

Good tools make huge volumes of trace data manageable.

If you're building your own application, you can make tracing an easy-to-use feature.

Tracing reduces chaos and simplifies performance optimization into a single process throughout your software life cycle.

Using trace data in code reviews leads to more efficient applications, and it bonds DBAs and developers into better partners.

Trace data can help you detect performance regressions before your users notice them.

Trace data helps everyone in your organization make better-informed business decisions.

Problem

Many Oracle adopters have no process in place for detecting performance problems before the application go-live milestone. The results can be ugly. Once I saw an application that was so inefficient it couldn't support its initial 20-user training rollout. This thing was supposed to serve thousands of users. It cost $30 million and had taken five years to write.

Companies that don't have a process for preventing performance problems have to figure out a process for solving them later. Do you really want to wager your career on expensive-sounding ideas derived from troubleshooting methods and tools you've created on the fly, through trial and error, under the crushing, unrelenting pressure of your users and leadership scrutinizing every step you take?

You know that solving problems earlier in the software life cycle is better for everybody, but how?

Plan

You need a method for troubleshooting performance problems that works both before and after go-live, for *any* performance problem. You need a method you can practice and get good at, that everyone on your team—your DBAs, your developers, your sysadmins, your architects, even your users and leadership—can learn.

Next, you need for your application to integrate with your method. You want it to be easy to collect the performance data you need, throughout your software life cycle: from testing your earliest prototypes, to evaluating the efficiency of new features, to baselining your well-behaved applications, to diagnosing your misbehaving applications.

Finally, you need software tools that help you manage, mine, and manipulate the performance data your application collects. You need tools that help everyone on your team understand how your programs spend your time.

Analysis

The method and tools for accomplishing these goals do exist. We learned in the early 2000s that the most valuable performance data for Oracle Database applications is *extended SQL trace* data. Tracing is a basic feature of every release of every Oracle edition. Used correctly, tracing is a high-detail data source that creates virtually no perceptible sacrifice in system performance.

Tracing is infeasible when you don't have good tools to manage, mine, and manipulate the huge volume of detail that it gives you. Our Method R Workbench software is the interface between your mountain of data and the job to be done.

The method, itself called Method R, keeps you focused on measuring the *response times* of the programs that are important to your business. With Method R, your process for *preventing* problems before go-live becomes identical to your process for *diagnosing* problems in production.

Solution

Accomplishing your goals begins with a few process changes:

1. Application designers and developers make your application easy to trace. This way, everyone on your team can have easy access to your programs' detailed performance data. If you've bought (not built) your application, it may not have all the performance measuring features you'd like built in, but any Oracle-based application can be traced.
2. If you have in-house programmers, they trace routinely during development to find inefficiencies and understand how their programs spend time.
3. Before any new application feature is promoted to production, a database performance specialist reviews the new code's trace data. If your programmers are in-house, the trace data for their code is analyzed at every code review.
4. Database operators routinely trace key application features in production. Routine tracing reveals tiny performance regressions before your users notice them, and performance baselines make it easier to diagnose problems later.
5. When you do encounter a production performance problem, use trace data to find where your programs spend their time. Even if you have to trace every program on your system for a few hours, Oracle and Method R Workbench can handle the volume.
6. Everyone—both technical and non—participates in the culture of *measuring over guessing*. For example, before your next upgrade, *measure* the response times of your key application features. *Predict* how the upgrade should change the response time for each feature. After the upgrade, *measure* response times again and *calibrate* your forecast. If your forecast was wrong, learn why. It's your fast-path to performance expertise.

Results

Every code review that finds an inefficiency is a problem that no user will ever experience. It's a programmer learning how to write better code, and it's an organization that is more intimate with the performance of the software it counts on for survival.

The rare problem that does slip through your code review filter, you'll deal with using the same method and data that you use in your code reviews. At first, you'll think, "I can't believe I didn't notice that; I'll never make *that* mistake again." And then you won't. Over time your performance (both your application's and yours personally) will get more and more bulletproof.

On your production system, your database operators will trace the performance of your application's key features even on good days when users aren't having problems, because you want to see any little jiggle of creeping performance problems. Keeping those trace files will make it easier to diagnose problems later.

Your company will spend less on hardware upgrades, because more efficient software runs faster on less hardware. When you do upgrade, you'll predict how much faster your different programs will be. "TPS reports will run in 72% less time. Pick-to-Ship will improve by only 8%; but it's already subsecond, so it doesn't matter." And you'll be right.

Your whole organization will achieve a continual intimacy with the performance of your application. You know where your performance risks are, and you know what it looks like when your application is creeping toward problematic behavior. You'll still find the occasional surprise, but most of your surprises will be in pre-production tests, and for the rare surprise that occurs in production, you'll have your method and your data ready.

Technology

Method R Workbench is easy-to-use, high-precision *Oracle time measurement software* for software development, code reviews, performance tests, concept proofs, hardware and software evaluations, upgrades, troubleshooting, and more—for Oracle developers, DBAs, and decision-makers in every phase of the software life cycle.

Method R Workbench 9

◇ Method R™
method-r.com
info@method-r.com