Making Your Application Easier to Diagnose

It's easy these days to mine interesting user experiences out of enormous trace data collections. But there's still value in being able to trace that one user or that one program, right when you need it.

by Cary Millsap

Making performance easier to measure and manager requires application features you may not have realized you can ask for.

An application that's easier to measure is less costly to operate and easier to optimize.

If you're writing your own application, it's easy to set identifying information that will make a program easy to trace later.

You can create easy-access performance observability into your application with just a little imagination and a few hours of application development effort.

Problem

Maybe you get a phone call, or maybe you overhear it in the lunchroom. Nancy claims that when she books an order on your sales order entry system, it's always slower then when anyone else does it. Is it true? Is the system really slower for Nancy? Maybe she just has tougher standards than everybody else. Or maybe she books bigger orders than everyone else. But maybe there's something wrong.

Step one is to separate the subjective from the objective. You'd like to trace Nancy's next BookOrder execution and three or four of her teammates' orders. Then you could do a side-by-side comparison about how the different executions spent time. But how in the world would you do that?

DBMS_MONITOR

An Oracle-based application doesn't need much extra code in it to make it easily traceable with the standard DBMS_MONITOR package distributed with every edition and release of the Oracle Database product.

DBMS_MONITOR is a little bit magic. It creates a *standing order* to trace a program in the future, without your having to be there to do it. For example, one procedure called CLIENT_ID_TRACE_ENABLE lets you specify that any time any program on your system identifies itself with a given client identifier, Oracle will automatically trace it. Even when that program runs at 3:14 a.m., Oracle will automatically trace it, without your having to wake up to do it. Oracle will trace every execution of such a program until you execute a CLIENT_ID_TRACE_DISABLE command to cancel the standing order.

Another procedure called SERV_MOD_ACT_TRACE_ENABLE does the same thing for programs that identify themselves by setting their service, module, and action names. You can see a program's service, module, action, and client identifier values in V$SESSION. The ENABLE and DISABLE procedures of DBMS_MONITOR behave as if you have a trigger on V$SESSION. But you don't.

Oracle Database provides everything you need to trace this way (you don't need any extra license options or packs), except for one thing:

You need your application programs to identify themselves so that DBMS_MONITOR *can find them.*

Instrumenting Your App

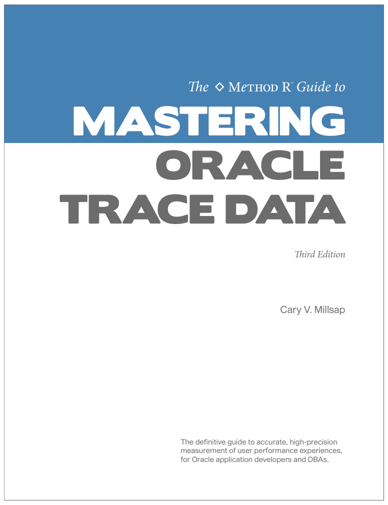
Some applications, like Oracle E-Business Suite, set unique names for many of their programs, making it easy to enable and disable tracing with DBMS_MONITOR.

Many applications that we visit in our field work don't make it easy to do targeted tracing. These applications are the reason we created the big-data features of our Method R Workbench application. For example, you can simply trace the entire Oracle instance for the time intervals you're interested in (sometimes many hours), and then use our Workbench to shred quickly through your trace files. Even if there are tens of thousands of them.

If you are building your own Oracle-based application, then you have the opportunity to make application tracing easy. Making it happen is also easy. How you do it depends on the language in which you write your business logic.

If you are writing your business logic in PL/SQL—and I hope you are, because that's the best way to maximize correctness, security, and performance—then use the DBMS_APPLICATION_INFO.SET_MODULE to set your program's module and action names, and DBMS_SESSION.SET_IDENTIFIER to set your program's client identifier value.

If you are writing your business logic in another language like Java, PHP, C#, or Python, the module, action, and client identifier fields are exposed as connection object properties that you can set or reset with just one line of code.


The Method R Guide to
MASTERING ORACLE TRACE DATA
Third Edition
Cary V. Millsap
The definitive guide to accurate, high-precision measurement of user performance experiences, for Oracle application developers and DBAs.

Either way you do it, the pseudocode looks like this:

```
set module, action, clientid
your business function's code path goes here
unset module, action, clientid
```

Performance, as a Feature

There's a lot more you could do. For example, you could have each application function consult a table that defines how often that function should be traced. You could implement rules like, "Trace only the first BookOrder transaction each hour," or "Trace only a randomly selected 5% of BookOrder executions." With just a little imagination, you can turn performance observability into a first-class application feature.

But you don't have to get fancy to make a lot of progress. Simply having your programs identify themselves with unique module/action name combinations is enough to give your database operators an easy way to trace any program they know the name of, now and forever. And having each of your programs set its client identifier value to a name that identifies who is running the program—that makes it easy for the database operator to trace Nancy's and three or four of her teammates' orders.

It's exactly the information you wanted, and it's not hard to do.

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. For full details about how to make your application easier to diagnose, order a copy of *The Method R Guide to Mastering Oracle Trace Data*.

Method R Workbench 9

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