Using the PL/SQL Hierarchical Performance Profiler

This is the readme that accompanies the zipfile of the presentation slides, the code that was used for the demonstation runs of the profiler, and the sets of links HTML files that are the visualizations of the profiler outout from each demonstation run.

The whole kit serves instead of a companion paper.

The kit can be downloaded from the Oracle Technical Network. Go here first:

www.oracle.com/technetwork/indexes/samplecode/plsql-sample-522110.html

Then select *plsql-hierarchical-profiler.zip*.

Now that you've unzipped *plsql-hierarchical-profiler.zip*, you'll notice that you have not just my presentation slides; you also have a set of representative HTML performance analysis reports.

These are for use during the presentations – in other words, it is something of a multimedia show.

The notes pages on the slides tell you which report to use at which point in the presentation; and the slides themselves tell you what to look out for in the reports.

You also have the programs that I used as test subjects for the performance analyses. The scripts will be self-explanatory for anyone used to using SQL*Plus and Oracle Database. You can use these to kick-start your use of the PL/SQL Hierarchical profiler. I wrote them to run on my Windows PC. You will need to update file paths if you want to run the experiments – on Windows or on Linux.

And, if you do re-run my experiments, and then use *plshprof* to format the HTML reports, you'll see that they aren't quite the same as the ones I used in the presentation. (The ones you make and the ones I used will be on separate directories.) I took the liberty of hand-editing the **.trc* files to remove the redundant `"USR".` qualification of the names and to change them into mixed case so as to improve the readability. I also hand-edited the derived **_pc.html* files produced by *plshprof*. I introduced lots of vertical whitespace between the block for each subprogram with its callers and callees so that when I navigated from block to block, each new view positioned the block at the same spot on the screen. All this was to make them more easily understandable in the context of a live presentation.

Demo_0

Some PL/SQL and some SQL. The dominant time is SQL time.

Callers_And_Callees

Contrived so that *Some_Subprogram* is called by *Caller_1*, *Caller_2*, and *Caller_3*; and calls *Callee_1*, *Callee_2*, and *Callee_3*. Intended simply to show how to navigate the reports.

Demo_1

Main() calls *p1()… p5()*. Some of these call *Helper()*.

Helper() is fast – there simply to reinforce how we understand the call graph.

One, *p3()*, stands out as especially long self-time. Fix it, re-run, and diff.

Shows that there were no significant changes to the other guys.

Demo_2

Main() calls *f1()… f5()*. No *Helper()* this time.

Each of the *f%()* functions is in its own PL/SQL unit and is fast; but each is called many times. *f5()* is called 100000 times. -> Run_1.

Move the implementation of *f5()* into the declare section of *Main()* and inline it. -> Run_2.

Demo_3

Main() calls *p1()… p5()*. Each of these calls *Helper()*. One of the callers, *p3()*, invokes *Helper()* with an actual argument that requests expensive tracing.

Bryn Llewellyn
Product Manager, Database Server Technologies Division, Oracle USA
400 Oracle Pkwy, Redwood City, CA 94065, USA
1-650-506-0224
bryn.llewellyn@oracle.com
www.oracle.com