# PL/SQL Tuning – Hierachical Profiler

**Radu Pârvu**
**Accenture**
**Finland**

**Keywords:**
Oracle, Accenture, database, profiler, PL/SQL, tuning

**Introduction**

Performance tuning overview
PL/SQL Profiler
PL/SQL Hierarchical Profiler
configuring the schema
collecting profiler data
understanding and interpreting the profiler data
using the plshprof utility
A few other optimizer techniques
Use the DBMS_UTILITY.GET_TIME Function
quick best practices (avoid procedural code, use triggers with care and only for small code, avoid type conversions etc.)

**Performance tuning overview**

General steps
Tune the access to code and data in SGA
SQL Optimize
Adjust the compiler optimization level
Apply best practices and standards
Check the execution profile
Algorithms
Use PL/SQL performance features
Validate the memory consumption. Done together with DBA. Implementation of PL/SQL features require sometimes a lot of PGA and maybe SGA. The application memory consumption have to be reasonable from system point of view.

**Main Question: What runs slowly?**

Profilers

DBMS_PROFILER
Legacy profiler
Switches on execution profiling at session level

DBMS_HPROF (hierarchical profiler)
Hierarchical profiler introduced on 11g
Rolls up results through the execution stack

**DBMS_PROFILER**

Setup
Install the program. Run the script as SYSDBA account:

```
$ORACLE_HOME/rdbms/admin/profload.sql
Verify. The statement below should not return error.
SQL> DESC DBMS_PROFILER
Create in own schema the tables populated by the profiler. Run:
$ORACLE_HOME/rdbms/admin/proftab.sql
```

three tables populated by DBMS_PROFILER will be created:
PLSQL_PROFILER_RUNS - Parent table of runs
PLSQL_PROFILER_UNITS - Program units executed in run
PLSQL_PROFILER_DATA -Profiling data for each line in a program unit

Usage
After the setup is completed, you start gathering profiling information for your application by writing code like this:

```
BEGIN
DBMS_PROFILER.START_PROFILER (
'Radus app' || TO_CHAR (SYSDATE, 'YYYYMMDD HH24:MI:SS')
);
radu_procedural_code;
DBMS_PROFILER.STOP_PROFILER;
END;
```

Usage
Find the profiler run of interest by checking the run_comment_column, make a note of the run_id:
select runid, run_owner, run_date, run_comment
from plsql_profiler_runs;
Next, enter the RUNID from the prior SQL statement. Oracle will place several lines of '' in the UNIT_OWNER column. This information is the overhead that Oracle incurred executing the code, not the code itself. We are not interested in this:

```
select runid, unit_number, unit_type, unit_owner, unit_name,
unit_timestamp
from plsql_profiler_units
where unit_owner <> ''
and runid = <collected run id>;
```

Next check the run data with a query like:

```
select pu.unit_name, pd.line#, pd.total_occur passes,
round(pd.total_time / 1000000000,5) total_time, us.text text
from plsql_profiler_data pd, plsql_profiler_units pu, user_source us
where pd.runid = <collected run id>
and pd.unit_number = <unit ids of interest>
and pd.runid = pu.runid
and pd.unit_number = pu.unit_number
and us.name = pu.unit_name
```

```
and us.line = pd.line#
and us.type in ('PACKAGE BODY','PROCEDURE','FUNCTION');
```

## DBMS_HPROF

Intro
Oracle DB 11g introduced a second profiling mechanism: DBMS_HPROF
Used to obtain the execution profile of PL/SQL code, organized by the distinct subprogram calls in your application
Unlike DBMS_PROFILER that record the time that your application spends within each subprogram, down to the execution time of each individual line of code, sometimes you also want to know how much time the application spends within a particular subprogram

Reports performance information about each subprogram in your application that is profiled, keeping SQL and PL/SQL execution times distinct.
The profiler tracks a wide variety of information, including:
the number of calls to the subprogram
the amount of time spent in that subprogram
the time spent in the subprogram's subtree (that is, in its descendent subprograms)
detailed parent children info

Components
Data collector
Turns hierarchical profiling on and off
The PL/SQL runtime engine writes the "raw" profiler output to the specified file.
Analyzer
Processes the raw profiler output and stores the results in hierarchical profiler tables from which profiler information can be displayed.

Usage
execute rights on the DBMS_HPROF package

WRITE privileges on the directory that you specify when you call DBMS_HPROF.START_PROFILING
create the three profiler tables (see details below)
call the DBMS_HPROF.START_PROFILING procedure to start the hierarchical profiler data collection in your session.

run your code long and repetitively enough
call the DBMS_HPROF.STOP_PROFILING procedure to terminate the gathering of profile data
analyze the contents and then run queries against the profiler tables

Create the three profiler tables
Run the dbmshptab.sql script (located in the rdbms/admin directory)
This script will create these three tables:
DBMSHP_RUNS: Top-level information about each run of the ANALYZE utility of DBMS_HPROF.
DBMSHP_FUNCTION_INFO: Detailed information about the execution of each subprogram profiled in a particular run of the ANALYZE utility

DBMSHP_PARENT_CHILD_INFO: Parentchild info for each subprog profiled in DBMSHP_FUNCTION_INFO.

Test the performance of my procedure
Start profiling:

```
BEGIN
DBMS_HPROF.start_profiling ('TEMP_DIR', ‘my_plsql_trace.txt');
END;
/
```

Call my PLSQL program:

```
BEGIN
my_proc (‘param1');
END;
/
```

- Stop profiling session:

```
BEGIN
DBMS_HPROF.stop_profiling;
END;
/
```

- Now the trace file is updated. It might be possible to open it and check it, but the most clever way would be to use the ANALYZE utility of the package DBMS_HPROF. This will take the trace file contents, tranforms it and places it into the three profiler tables. **Important:** it will return a run number to be used when querying those tables.

**Interpreting the profiler data**

- Analyze:

```
BEGIN
DBMS_OUTPUT.PUT_LINE (
DBMS_HPROF.ANALYZE ('TEMP_DIR', 'my_plsql_trace.txt'));
END;
/
```

**Generate simple HTML report**

- Achieved by running the plshprof command-line utility
- Located in the directory $ORACLE_HOME/bin/
- Generates simple HTML reports from either one or two raw profiler output files
- For an example of a raw profiler output file, see the section titled “Collecting Profile Data” in the Oracle Database Advanced Application Developer’s Guide
- We can use the generated HTML reports in any browser

Your contact details at the end of your article.

**Contact address:**

**Name**
Company
Address
Postal code, city

Phone: +49(0)12-3456789
Fax: +49(0)12-3456788
Email Your@emailaddress
Internet: Address