Profiling Oracle: How it Works

Cary Millsap
Hotsos Enterprises, Ltd.
Suncoast Oracle Users Group / Tampa, Florida
6:00pm–8:00pm Thursday 30 March 2006

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

hotsos SM
www.hotsos.com

Oracle. Performance. Now.

We can show you how to do
everything in this presentation.
Or we can do it for you.

Agenda

- Introduction
- The role of profilers
- Profiling Oracle
- Case study
- Discussion

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

So there will be no confusion...

- This presentation does describe
  - Why you use a profiler
  - How you use profile data
- This presentation does not describe
  - How to create a profile from raw trace data
- We can do that…
  - But it takes more than an hour
  - Visit www.hotsos.com for more information

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

The role of profilers

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Performance analysis is the study of how long something takes.

- Imagine…
  - Q: “How long does it take to fly on AA from DFW to TPA?”
  - A: “Mean fuel consumption in 2004 was *F* lbs/flight”
- This isn’t an answer!
  - Wrong unit of measure
  - Includes stuff you don’t want (LAX-SIN, HNL-OGG, …)
  - Different aircraft types with different consumption rates

What you need is “How long does it take to fly on AA from DFW to TPA?!”

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

In Oracle, performance analysis has become the study of just about everything but response time.

- From everyday life in Oracle…
  - Q: "Why does *P* take so long?"
  - A: "Here are your hit ratios, your utilization rates, …"
- How long? Why?
  - You cannot tell from these reports

Oracle analysis tools are medieval compared to the tools and methods that software developers use

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

A profiler is a tool that reports a useful decomposition of a computer program's response time.

- Nothing new
  - Knuth described a FORTRAN profiler in 1971
  - The GNU *gprof* profiler has been around since 1988
  - …
- Profilers are indispensable application development tools
  - Diagnosis
  - Instruction
  - Debugging

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Here's how profiling fits into your life…

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

The simplest profile is the flat profile, which shows response time decomposed by subroutine call.

```
$ gprof
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  us/call  us/call  name
 60.37      0.49     0.49 62135400     0.01     0.01  step
 39.63      0.82     0.33   499999     0.65     1.64  nseq
```

- Response time is 0.82 seconds
- 60.37% is consumed by 62,135,400 calls to *step*
- *nseq* takes longer per call, but contributes less total time
- …

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Another type of profile is the call graph, which shows hierarchical interactions between subroutines.

```
                Call graph (explanation follows)
…

index % time    self  children    called     name
                0.33    0.49  499999/499999      main [2]
[1]    100.0    0.33    0.49  499999         nseq [1]
                0.49    0.00 62135400/62135400     step [3]
-----------------------------------------------
                                                   <spontaneous>
[2]    100.0    0.00    0.82                 main [2]
                0.33    0.49  499999/499999      nseq [1]
-----------------------------------------------
                0.49    0.00 62135400/62135400     nseq [1]
[3]     60.4    0.49    0.00 62135400         step [3]
-----------------------------------------------
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Profiling Oracle

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

You can use *gprof* on <u>your</u> code, but not Oracle's.

- To use *gprof*

```
$ gcc -pg myprogram.c
$ a.out
$ gprof
```

- Works great for application code you're writing
- The problem…
  - You can't compile *oracle.c*

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

The Oracle kernel has a feature built in that allows you to profile its internal operations as well.

- Extended SQL trace
  - The event formerly known as "10046 level 12"
  - Works great with versions 7, 8, and 9; even better in 10
- Gives you everything you need to create…
  - Flat profile
    - …by Oracle subroutine (timed syscalls ∪ dbcalls ∪ …)
    - …by database call
  - Call graph
    - …by statement (SQL or PL/SQL)
  - Lots more

This is a <u>huge</u> capability

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Version 10 example: Using extended SQL trace…

```
$ sqlplus
…
SQL> show parameter…
NAME                          TYPE        VALUE
----------------------------- ----------- -------------------------------
timed_statistics              boolean     TRUE
max_dump_file_size            string      UNLIMITED
user_dump_dest                string      /u01/app/oracle/admin/v10/udump
…
SQL> exec dbms_monitor.session_trace_enable(null,null,true,true);
SQL> select 'hello' from dual;
SQL> exec dbms_monitor.session_trace_disable(null,null);
…

$ ls -lt $UDUMP
total 248
-rw-r-----  1 oracle   oinstall      2396 Jan 17 11:18 v10_ora_5286.trc
…
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Version 9 example: Using extended SQL trace…

```
$ sqlplus
…
SQL> show parameter…
NAME                          TYPE        VALUE
----------------------------- ----------- -------------------------------
timed_statistics              boolean     TRUE
max_dump_file_size            string      UNLIMITED
user_dump_dest                string      c:\Oracle\admin\v92\udump
…
SQL> exec dbms_support.start_trace(true,true);
SQL> select 'hello' from dual;
SQL> exec dbms_support.stop_trace;
…

$ ls -lt %udump%
total 44
-rw-rw-rw-  1 user     group         4083 Jan 17 02:29 v92_ora_2272.trc
…
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

hotsos
www.hotsos.com

The trace file contains everything you need to create a flat profile by Oracle subroutine.

```
Oracle subroutine              Duration (secs)   # Calls    Dur/call
------------------------------ ---------------- --------- -----------
SQL*Net message from client     984.010   49.6%    95,161    0.010340
SQL*Net more data from client   418.820   21.1%     3,345    0.125208
db file sequential read         279.340   14.1%    45,084    0.006196
CPU service, EXEC               136.880    6.9%    67,888    0.002016
CPU service, PARSE               74.490    3.8%    10,098    0.007377
CPU service, FETCH               37.320    1.9%    57,217    0.000652
unaccounted-for                  27.720    1.4%         1   27.720000
latch free                       23.690    1.2%    34,695    0.000683
log file sync                     1.090    0.1%       506    0.002154
SQL*Net more data to client       0.830    0.0%    15,982    0.000052
log file switch completion        0.280    0.0%         3    0.093333
enqueue                           0.250    0.0%       106    0.002358
…
------------------------------ ---------------- --------- -----------
Total response time           1,985.190  100.0%
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

hotsos
www.hotsos.com

The trace file contains everything you need to create a call graph by SQL or PL/SQL statement.

```
                                     Duration of    Duration incl.
Cursor statement                     self (secs)   children (secs)
----------------------------------- ---------------- ---------------
/* REGULAR_PAY*/DECLARE LERMT VA…        6.620   0.3%   67.270   3.4%
|  SELECT 'Y' FROM PAY_ELEMENT_ENTR…    35.020   1.8%   35.020   1.8%
|  SELECT DECODE(COUNT(PRR.RUN_RESU…    14.740   0.7%   14.740   0.7%
|  SELECT DECODE(COUNT(RRS.RUN_RESU…     9.090   0.5%    9.090   0.5%
|  SELECT DECODE(NVL(TO_CHAR(PDS.AC…     1.110   0.1%    1.110   0.1%
|  SELECT START_DATE,END_DATE FROM …     0.690   0.0%    0.690   0.0%
update pay_person_latest_balance…       66.650   3.4%   66.650   3.4%
select ASSBAL.defined_balance_id…       64.580   3.3%   64.580   3.3%
update pay_assignment_latest_bal…       61.140   3.1%   61.140   3.1%
select 1 into :b0 from sys.dual …       36.470   1.8%   36.470   1.8%
/* REGULAR_EARNINGS*/DECLAREL_ER…        8.080   0.4%   35.670   1.8%
|  SELECT DECODE(COUNT(PRR.RUN_RESU…    26.280   1.3%   26.280   1.3%
…
----------------------------------- ---------------- ---------------
Total response time                 1,985.190 100.0%
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Case study

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Baseline case: inserting is too slow.

- Each process inserts 5,000 rows
- 2 concurrent processes
- 1-CPU Windows XP
- Oracle 9.2.0.4
- Connection configuration

```
v92 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = CVM-LAP02)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = v92.hotsos)
    )
  )
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Baseline response time profile…

v92_ora_2020.trc

Oracle subroutine	Duration (secs)		# Calls	Dur/call
unaccounted-for	7.292	60.4%	10,255	0.000711
SQL*Net message from client	2.371	19.6%	20,007	0.000119
CPU service, PARSE calls	1.843	15.3%	5,040	0.000366
CPU service, EXEC calls	0.411	3.4%	5,061	0.000081
SQL*Net message to client	0.066	0.5%	20,007	0.000003
db file sequential read	0.055	0.5%	2	0.027650
log file sync	0.017	0.1%	3	0.005714
CPU service, FETCH calls	0.010	0.1%	152	0.000066
latch free	0.006	0.0%	19	0.000315
Total	12.071	100.0%		

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Typical first questions about the profile…

- What is “unaccounted-for”?
- Isn’t *SQL*Net message from client* supposed to be “idle” time?
- Why does this thing parse almost 4.5× longer than it inserts?

If the point is to insert 5,000 rows, then why does inserting consume only 3.5% of total response time?

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

The call graph shows that almost all the time is consumed by one statement.

```
                                          Duration of    Duration incl.
Cursor statement                          self (secs)   children (secs)
---------------------------------------- ---------------- ---------------
insert into t values (1, lpad('1…         11.309  93.7%   11.392  94.4%
|  select u.name,o.name, t.update$,…       0.063   0.5%    0.063   0.5%
|  select file# from file$ where ts…       0.017   0.1%    0.017   0.1%
|  update tsq$ set blocks=:3,maxblo…       0.002   0.0%    0.002   0.0%
[[ synthetic parent ]]                     0.000   0.0%    0.582   4.8%
|  select u.name, o.name, trigger$.…       0.472   3.9%    0.493   4.1%
|  |  select ts#,file#,block#,nvl(bobj…    0.015   0.1%    0.015   0.1%
|  |  select i.obj#,i.ts#,i.file#,i.bl…    0.002   0.0%    0.002   0.0%
|  |  select name,intcol#,segcol#,type…    0.002   0.0%    0.002   0.0%
|  |  select pos#,intcol#,col#,spare1,…    0.001   0.0%    0.001   0.0%
|  |  select type#,blocks,extents,mine…    0.001   0.0%    0.001   0.0%
|  select order#,columns,types from…       0.002   0.0%    0.040   0.3%
… (35 other statements have been elided)
----------------------------------------  ------- ------  ------- ------
Total                                     12.071 100.0%
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

The database call profile shows that the preponderance of the response time is spent <u>between</u> db calls.

```
insert into t values (1, lpad('1',20))

Oracle hash value: 2581399381
 Statement re-use: 5,000 similar but distinct texts
    Response time: 11.309 seconds (93.7% of task total 12.071 seconds)

                      -----Duration (seconds)-----
Database call            Elapsed      CPU    Other   Calls   Rows
--------------------  -------------  -----  ------  ------  -----
Between-calls          6.518  57.6%  0.000   6.518       0      0
PARSE                  3.801  33.6%  1.843   1.958   5,000      0
EXEC                   0.991   8.8%  0.411   0.580   5,000  5,000
--------------------  -------------  -----  ------  ------  -----
Total                 11.309 100.0%  2.253   9.056  10,000  5,000
--------------------  -------------  -----  ------  ------  -----
Total per EXEC         0.002   0.0%  0.000   0.002       2      1
Total per row          0.002   0.0%  0.000   0.002       2      1
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

The baseline program executed far too many parse calls.

- Old code behavior…

```
for each row {
     $sql = "insert into t values ($v1, lpad('$v2',20))";
     $c = parse($sql);
     exec($c);
}
```

- New code…

```
❶$c = parse("insert into t values (❷?, lpad(❷?,20))");
for each row {
     exec($c, ❸$v1, ❸$v2);
}
```

The rewrite should eliminate 4,999 parse calls

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

The results are spectacular (before).

v92_ora_2020.trc

Oracle subroutine	Duration (secs)		# Calls	Dur/call
unaccounted-for	7.292	60.4%	10,255	0.000711
SQL*Net message from client	2.371	19.6%	20,007	0.000119
CPU service, PARSE calls	1.843	15.3%	5,040	0.000366
CPU service, EXEC calls	0.411	3.4%	5,061	0.000081
SQL*Net message to client	0.066	0.5%	20,007	0.000003
db file sequential read	0.055	0.5%	2	0.027650
log file sync	0.017	0.1%	3	0.005714
CPU service, FETCH calls	0.010	0.1%	152	0.000066
latch free	0.006	0.0%	19	0.000315
Total	12.071	100.0%		

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

The results are spectacular (after).

```
v92_ora_2588.trc

Oracle subroutine              Duration (secs)    # Calls     Dur/call
----------------------------  -----------------  ---------  -----------
unaccounted-for                    0.631  41.2%      5,032     0.000125
SQL*Net message from client        0.393  25.7%      5,010     0.000078
CPU service, PARSE calls           0.090   5.9%         11     0.008194
CPU service, EXEC calls            0.381  24.9%      5,011     0.000076
SQL*Net message to client          0.008   0.5%      5,010     0.000002

log file sync                      0.027   1.8%          1     0.027396
CPU service, FETCH calls           0.000   0.0%          9     0.000002

----------------------------  -----------------  ---------  -----------
Total                              1.530 100.0%
```


Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Why are the results so much better than we expected?

- Eliminating dbcalls has several better-than-linear positive effects
  - We got the `CPU service, PARSE` estimate right
  - Eliminating ~5,000 parse calls eliminated ~15,000 `SQL*Net message from client` calls
  - Preemption time decreased better than linearly!

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Eliminating unnecessary workload often result in better-than-linear performance improvement.

- Eliminating work…
  - Saves top-line response time
  - Eliminates dependant work
  - Reduces queue lengths
    - Makes remaining calls faster
    - Helps everyone

Eliminating unnecessary work reverses exponential performance degradation

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

Recap

- A profiler tells you exactly where a task's time has gone
- Oracle emits everything you need to begin profiling
- Profilers give two capabilities you're not accustomed to having
  - "What happened?" becomes very easy
  - "What if?" becomes very easy

Once you have these capabilities, it's hard to imagine life without them

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

References

Bentley, J. 1988. *More Programming Pearls*. Reading MA: Addison-Wesley.
This book provided the stimulus for one of the earliest prototypes of the Hotsos Profiler's resource profile-based output format.
Dowd, K. 1993. *High Performance Computing*. Sebastopol CA: O'Reilly.
This is a good book for programmers who need to optimize the performance of their applications.
Gough, B. J. 2004. *An Introduction to GCC—for the GNU compilers gcc and g++*. Network Theory Ltd.
This is the source of the *collatz.c* GNU *gprof* example used in this presentation.
Hotsos Enterprises, Ltd. 2005. Hotsos Profiler at www.hotsos.com
The Hotsos Profiler meets all the profiling specifications described in this document.
Knuth, D. E. 1971. "Empirical study of FORTRAN programs" in *Software—Practice and Experience*, Apr/Jun 1971, Vol. 1, No. 2, pp105–133.
This is the earliest reference to computer application profilers that I know of.
Millsap, C. 2003. "Oracle operational timing data" at *www.hotsos.com*.
This is a brief article outlining the operational timing features of the Oracle kernel.
Millsap, C.; Holt, J. 2003. *Optimizing Oracle Performance*. Sebastopol CA: O'Reilly.
This book provides a full description of Method R, a detailed reference for Oracle's extended SQL trace facility, an introduction to queueing theory for the Oracle practitioner, and a set of worked performance improvement example cases.
Millsap, C. 2004. "Diagnosing performance problems" in *Oracle Magazine*, Jan/Feb 2004, pp68–70.
This article provides insight into the motives and mechanics of extended SQL trace.
Millsap, C. 2004. "How to activate extended SQL trace" at *www.hotsos.com*.
This article provides details about the mechanics of activating extended SQL trace.
Nørgaard, M.; Ensor, D.; Gorman, T.; Hailey, K.; Kolk, A.; Lewis, J.; McDonald, C.; Millsap, C.; Morle, J.; Ruthven, D.; Vaidyanatha, G. 2004. *Oracle Insights: Tales of the Oak Table*. Berkeley CA: Apress.
I wrote Chapter 5 of this book, which details the history and evolution of Oracle's extended SQL trace feature.
Oracle Corporation. Various technical resources:
`DBMS_MONITOR` package: http://download-west.oracle.com/docs/cd/B14117_01/appdev.101/b10802/d_monito.htm#ARPLS091.
`DBMS_SUPPORT` package: Oracle Metalink doc id Note:62294.1.
New version 10 OCI attributes: http://download-west.oracle.com/docs/cd/B14117_01/appdev.101/b10779/oci08sca.htm#452699

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.

A game…

- How would each problem show up in a profile?
  1. Program parsed inside a loop
  2. Program is efficient but can't get enough CPU to run fast
  3. SQL used an inefficient execution plan
  4. Other program's SQL used an inefficient execution plan
  5. Table had too many extents
  6. Disk subsystem was too busy
  7. Client code path consumed too much time
  8. Other program locked my table
  9. Too many programs update same block
  10. Shared pool was too small

Copyright © 1999–2006 by Hotsos Enterprises, Ltd.