Tracing Bind Variables and Waits

Christian Antognini
Trivadis AG
Zürich, Switzerland

Introduction

A common starting point in tuning an application is looking for poor SQL statements. The easiest way to do this is to enable SQL_TRACE with TIMED_STATISTICS and to analyse the generated trace files with TKPROF.
In most situations the TKPROF output is OK, but sometimes additional information will help understanding the problem and consequently make better tuning decisions.
The goal of this article is to show how to use an Oracle event to obtain more information in the trace files and how to interpret those outputs.

Events

What are Events?

An event is a special item used by Oracle developers that can be activated to do one of the following:

- to change a behaviour (activating/de-activating a feature)
- to test or simulate a corruption/crash
- to collect trace or debug information

For some events, different levels can be used. Each level will have its own behaviour.

When to set an Event?

You should only set an event directed to do so by Oracle Support or if you know and understand what the event is going to change.

Where to Find Events?

Events are not documented.
On some UNIX platforms they can be found in the file containing the ORA errors, notably $ORACLE_HOME/rdbms/mesg/oraus.msg. The range 10000 ... 10999 is reserved for them.

How to Set an Event?

There are many ways to set events and some of them are event specific. In the next section some examples are given.

Other Important Considerations

Events and their levels can change between Oracle releases, thus you must be sure to set a valid one.

If a problem occurs when an event is set then it is worth seeing if the same problem can be reproduced without the event set. This is because events can enable code specific paths which are not normally used.
All event set in the init.ora should be removed prior to any upgrade.

Extended SQL Trace

The event 10046 is used to enable SQL trace.

Levels

The following table describes the different levels and their behavior.

Level	Description
1	standard SQL trace functionality
4	as level 1 plus tracing of bind variables
8	as level 1 plus tracing of wait events
12	as level 1 plus tracing of bind variable and waits events

Level 1

The trace files are the same has those generated by the standard SQL trace (e.g. by command ALTER SESSION SET SQL_TRACE = TRUE).

Level 4

Some information about the bind variables is also written into the trace file.
For example if you have the statement:

```
select ID
from CI_APPLICATION_SYSTEMS
where NAME = :name and VERSION = :ver
```

for each execution you can know which values were used to execute it, for example:

```
exec    bind   value
------- ------ --------------------
      1      1 NULL
             2 0
      2      1 "PORTAL"
             2 1
```

For the first execution

```
:name = NULL
:ver = 0
```

and for the second

```
:name = "PORTAL"
:ver = 1
```

For each variable you can also have its type and precision.

Level 8

Some information about the waits occurred during the execution of the statement is written to the trace file. For example if you have the statement:

```
DELETE
FROM RM_DEFERRED_CHECKS
```

you can know on which waits the statement occurred, for example:

```
elapsed  count operation
------- ------ -------------------------------------------------------------
   2.50    148 db file scattered read
```

```
                D:\ORACLE\ORADATA\A815\DESIGNER01.DBF 736 blocks
  0.08        4 db file sequential read
                D:\ORACLE\ORADATA\A815\DESIGNER01.DBF 4 blocks
```

The statement has waited 2,58 seconds because the backend (a dedicated or shared server process) had to read 780 blocks from the disk to the SGA.

How to Set Event 10046

There are many ways to set this event. Here are some examples to set event 10046 at level 4:

- at instance level each time the instance is started (entry in init.ora)
  EVENT = "10046 trace name context forever, level 4"
- at instance level when the instance is running (only with 8i)
  alter system set events '10046 trace name context forever, level 4'
- at session level in the current session
  alter session set events '10046 trace name context forever, level 4'
  exec dbms_system.set_ev(<sid>, <serial#>, 10046, 4, '')
- at session level in another session
  exec dbms_system.set_ev(<sid>, <serial#>, 10046, 4, '')

All these examples works without any option (they use standard packages/commands). Another way to set this event is via the DBMS_SUPPORT package. The next section will give more information about it.

DBMS_SUPPORT Package

This package is used to set event 10046. It was developed by Oracle Support to give a simple and useful interface. It is only distributed on few platforms/versions (the files are named dbmssupp.sql and prvtsupp.plb), but it can be requested from Oracle Support. Contrary to its name, this package is not supported by Oracle (i.e. you are responsible to use it).

Version 1.0 contains the following functions:

mysid returns number
This function returns the SID of the current session

start_trace(waits in boolean, binds in boolean)
This procedure starts tracing in the current session.

stop_trace
This procedure stops trace in the current session.

start_trace_in_session(sid in number, serial in number, waits in boolean, binds in boolean)
This procedure starts tracing in the given session.

stop_trace_in_session(sid in number, serial in number)
This procedure stops tracing in the given session.

package_version returns varchar2
This function returns the package's version.

According to function *package_version* it requires version 7.2 to 8.0.5, but I also used it on 8.1.5 without problems.

Extended SQL Trace Files Analysis

If you try to analyze an extended SQL trace file (i.e. a file generated with a level greater than 1) with TKPROF you will see that bind variables and waits are skipped, no information will be generated.
To interpret the trace file you have to read Oracle Support note 39817.1. It is a short reference about the trace file format. Of course reading the trace files is not a solution.

TVD$XTAT

To solve this problem I wrote in PL/SQL my own analysis tool.
The trace file is parsed and the information loaded into the database with the command:

```
tvd$xtat.load(<file name>, <input mode>)
```

The input file can be red via a DIRECTORY object (TVD$XTAT_DIR) or the UTL_FILE package. Once the information are loaded into the database they can be analyzed with the command:

```
tvd$xtat.analyze(<output file>, <sys>, <num binds>)
```

The output file can be written to the standard output or via the UTL_FILE package into a file. Statement for user SYS can be listed or not. The maximum number of listed bind variable sets can be given (to each execution correspond a set of bind variables).
For each statement an output similar to TKPROF plus the section for bind variables and waits is generated.

```
SELECT ALL
  K.CONSTRAINT_REFERENCE PK_CONSTRAINT_REFERENCE,
  C.NAME PK_COLUMN_NAME
FROM CI_KEY_COMPONENTS K, CI_COLUMNS C
  WHERE ( K.COLUMN_REFERENCE = C.ID ) AND ( :PK_ID = K.CONSTRAINT_REFERENCE)
ORDER BY K.SEQUENCE_NUMBER

exec    bind   value
------- ------ --------------------------------------------------------------
      1      1 28122
      2      1 28751
      3      1 28320
      4      1 29059
      5      1 29008
      6      1 28779
      7      1 28732
      8      1 28982
      9      1 28526
     10      1 28277

call     count       cpu    elapsed       disk      query    current     rows
------- ------ --------- ---------- ---------- ---------- ---------- --------
Parse        7      0.35       0.56         14         50          0        0
Execute     62      0.24       0.54          0          0          0        0
Fetch       62      0.08       0.47         15        530          0       84
------- ------ --------- ---------- ---------- ---------- ---------- --------
total      131      0.67       1.57         29        580          0       84

elapsed  count operation
------- ------ --------------------------------------------------------------
   0.65     66 SQL*Net message from client
```

```
   0.34     15 db file sequential read
                D:\ORACLE\ORADATA\A815\DESIGNER01.DBF 15 blocks
   0.01     66 SQL*Net message to client

Misses in library cache during parse: 1
Misses in library cache during execute: 0
Optimizer goal: CHOOSE
Parsing user id: 23 (CHA)
```

The summary section is different from TKPROF. In fact no recursive statement totals are listed. Only the totals for each user and for the waits is generated.

```
OVERALL TOTALS FOR USER 0 (SYS)

call     count       cpu    elapsed       disk      query    current     rows
------- ------ --------- ---------- ---------- ---------- ---------- --------
Parse      637      3.45       4.09         24        170          5        0
Execute    822      5.97       6.89         10         74         28        2
Fetch     2455      4.89      23.56       1561      76796         44     2075
------- ------ --------- ---------- ---------- ---------- ---------- --------
total     3914     14.31      34.54       1595      77040         77     2077
OVERALL TOTALS FOR USER 23 (CHA)

call     count       cpu    elapsed       disk      query    current     rows
------- ------ --------- ---------- ---------- ---------- ---------- --------
Parse      172      8.14      15.71        267       1409          5        0
Execute    701     15.55      32.68       2272      76343        130        3
Fetch      954      2.41       9.62        555      12610        108     1468
------- ------ --------- ---------- ---------- ---------- ---------- --------
total     1827     26.10      58.01       3094      90362        243     1471

OVERALL TOTALS FOR WAITS

elapsed  count operation
------- ------ ------------------------------------------------------------
  68.45   1390 SQL*Net message from client
  25.13   2151 db file sequential read
               D:\ORACLE\ORADATA\A815\SYSTEM01.DBF 1710 blocks
               D:\ORACLE\ORADATA\A815\DESIGNER01.DBF 439 blocks
               D:\ORACLE\ORADATA\A815\ROLLBACKBIG.DBF 2 blocks
   2.36    194 db file scattered read
               D:\ORACLE\ORADATA\A815\DESIGNER01.DBF 966 blocks
   0.77     10 control file sequential read
   0.60      2 checkpoint completed
   0.19     49 file open
   0.09      4 file identify
   0.04   1390 SQL*Net message to client
   0.03     30 direct path read
   0.03      1 log file sync
```

If you are interested, send me an email. I will send you a copy. The script is also available at www.trivadis.com

Conclusion

Unfortunately, once again Oracle has "hidden" from us another interesting functionality. As a consequence there is no support given for it. In addition, TKPROF does not support the new trace files' entries and forces the users to develop some new utilities (Oracle Support has developed some tools but they are not distributed).

In any case the feature is interesting, and was worth the time spent to be able to use it effectively.