Introducing Oracle Extended Tracing

Robin East
Xense – Documentum Performance Consulting

Introduction

You have a performance problem – users' response times have deteriorated, DMCL tracing shows the problem is in the database but where? If you are using Oracle then you are in luck. This article describes a little known feature of Oracle tracing that allows you to identify exactly where Oracle is spending time executing users' queries. In addition I'll show you the easiest way of starting a trace session for a selected user's session.

Oracle Extended Trace

Many readers will be familiar with the standard Oracle tracing facility. It can be invoked by executing the command:

```
SQL> ALTER SESSION SET SQL_TRACE = T
```

The result is a trace file that contains details of all the database calls issued by the Content Server. The standard trace file contains a record of the CPU execution time and the elapsed time for each database call. After processing the trace file with TKPROF output similar to Figure 1 is obtained.

```
INSERT  INTO dm2.dm_sysobject_r
VALUES
('0900022880006500',   -16, NULL,'hotel', NULL, NULL, NULL, NULL, NULL,
  NULL, NULL, NULL, NULL, NULL, NULL, NULL)


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.01       1.53          0          1          6           1
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.01       1.53          0          1          6           1
```

Figure 1 Standard TKPROF output

Sometimes this information is sufficient to determine the cause of the performance problem. However in situations, such as the above, where the elapsed time is significantly greater than the CPU service time some essential information is missing. In this example the query took 1.53 seconds to execute however only 0.01 seconds of CPU time was used. Where did the other 1.52 seconds go? Maybe it was I/O, network delays or some other contention point in the database.

The extended SQL trace (level 8 or 12) augments the standard trace information by outputting timing data for wait events. Wait events are issued from instrumented sections of the Oracle kernel code that contain non-CPU consuming instructions.

For example during query execution if a required data block is not found in the buffer cache it must be read from disk. To do this the query execution code issues an I/O read request to the operating system. Whilst the I/O request is being serviced the query execution code 'sleeps' yielding the CPU for use by other threads (Windows) or processes (Unix). When the data block has been read in from disk, query execution continues. The Oracle kernel code writes information to the trace file describing the type of wait event (for data block I/O reads usually 'db file sequential read ' or 'db file scattered read') along with other information including the elapsed time between the start and finish of the instrumented section. There are similar instrumented sections for waits on locks, latching, logging operations and many more.

Using TKPROF from Oracle version 9 and above we can now improve on the above output using the waits=yes option:

```
INSERT  INTO dm2.dm_sysobject_r
VALUES
('0900022880006500',   -16, NULL,'hotel', NULL, NULL, NULL, NULL, NULL,
  NULL, NULL, NULL, NULL, NULL, NULL, NULL)


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.01       1.53          0          1          6           1
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.01       1.53          0          1          6           1

...

Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  log buffer space                                6        1.00          1.52
  SQL*Net message to client                       2        0.00          0.00
  SQL*Net message from client                     2        0.00          0.00
```

Figure 2 TKPROF output on level 8 extended trace with waits=yes

It's now easy to see where the time went, a non-CPU consuming instrumented section called 'log buffer space'. So what is a 'log buffer space' event? The place to look is in the Oracle Reference Manual (freely available on the web at http://otn.oracle.com) in the appendix on Wait Events, where it states:

Waiting for space in the log buffer because the session is writing data into the log buffer faster than LGWR can write it out. Consider making the log buffer bigger if it is small, or moving the log files to faster disks such as striped disks.

We've located the problem fast and without having to guess – that is some improvement over the traditional methods of resolving performance problems.

How to set extended tracing for Documentum users

If you are using SQL Plus then you can issue the following statement to activate extended SQL tracing for your own session:

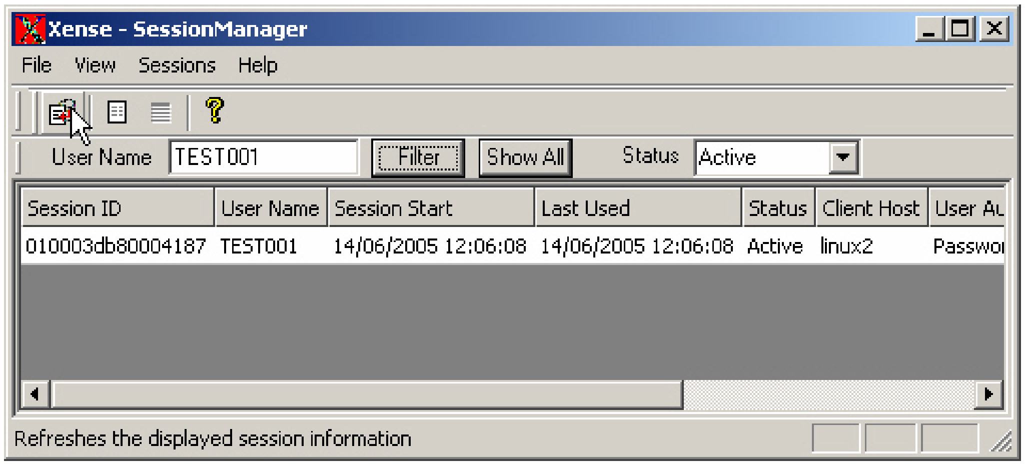
```
ALTER SESSION SET EVENTS '10046 trace name context forever,level 8'
```

This is often useful when performance tuning a particular DQL query and you already know what SQL has been generated. A similar command can be issued from DQL if you are a superuser (all quotes are single-quotes):

```
EXECUTE exec_sql with query = 'alter session set events ''10046 trace
name context forever,level 8'''
```

However, starting a trace for a user session without superuser rights is more difficult. My company Xense provides a Content Server session monitoring tool called Session Manager which provides a function to initiate extended SQL tracing. The tool is free and is available at http://www.xense.co.uk/resources_xsm_start.htm .

Once you have downloaded and installed Session Manager you can start the application, which should give you a screen similar to the following (I have used the User Name filter to restrict the display to just the user I am interested in):

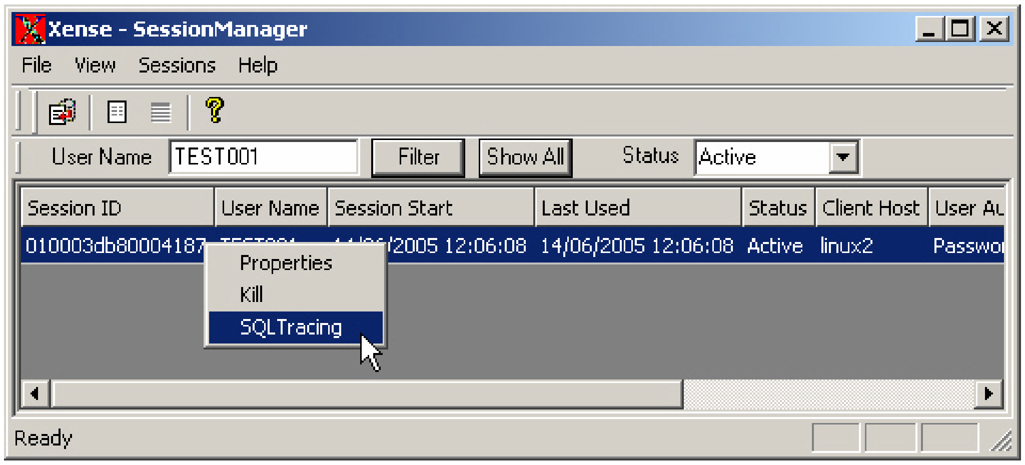


User Name: TEST001
Status: Active

Session ID	User Name	Session Start	Last Used	Status	Client Host	User Au
010003db80004187	TEST001	14/06/2005 12:06:08	14/06/2005 12:06:08	Active	linux2	Passwo

Refreshes the displayed session information

Now right-clicking on the selected session gives the following pop-up menu:

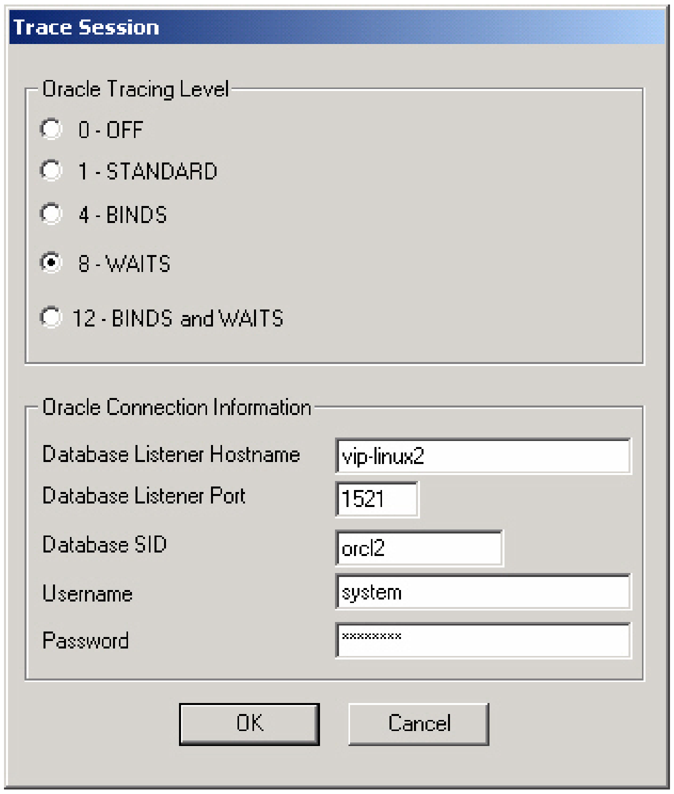


Session ID	User Name	Session Start	Last Used	Status	Client Host	User Au
010003db80004187	TEST001	14/06/2005 12:06:08	14/06/2005 12:06:08	Active	linux2	Passwo

User Name: TEST001
Status: Active

Select SQL Tracing and the Trace Session dialog is displayed.

Complete the following dialog (Session Manager 'remembers' the information for the duration of your session so you only have to enter the information once):


Trace Session

Oracle Tracing Level

( ) 0 - OFF
( ) 1 - STANDARD
( ) 4 - BINDS
[✓ 8 - WAITS]
( ) 12 - BINDS and WAITS

Oracle Connection Information

Database Listener Hostname	vip-linux2
Database Listener Port	1521
Database SID	orcl2
Username	system
Password	********

Use the information appropriate for your environment as follows:

Database Listener Hostname	:	This is the hostname of the listener, usually the same machine that hosts the database
Database Listener Port	:	Most listeners are configured on 1521
Database SID	:	The oracle SID for the database hosting the docbase
Username	:	The Oracle username with permission to access the tracing facility (see installation instructions for details)
Password	:	Password for the user above

Click OK to start the trace.

All SQL statements generated from the user's Documentum session are now sent to the Oracle trace file.