# Art and Craft of Tracing

**Arup Nanda**

***Longtime Oracle DBA***

## Agenda

*"My session or application is slow, or not acceptable. Can you find out why?"*

- What is tracing
- Types of tracing
- Tracing in a current session
- Tools to analyze tracefiles
- Tracing a different session
- Tracing for future sessions
- Client Identifier and Client ID
- Tracing in RAC
- Consolidating tracefiles

## What is Tracing?

- Execution plan tracing
- Enables inner workings of the session
- Queries executed
  - Including recursive queries
- Details captured
  - Execution plans
  - Time spent
  - Rows affected
  - Parses, etc.
- Other type of trace: 10053 (CBO decision)

## Simple Tracing

- All relevant information

```
SQL> alter session set sql_trace = true;
```

- Must have “alter session” privilege
- Creates a tracefile in
  - ≤ 10g – user_dump_dest directory
  - ≥ 11g – ADR: <OracleBase>\diag\rdbms\<DBName>\<OracleSID>\trace
- Named `<OracleSID>_ora_<spid>.trc`
- Put a phrase in the name

```
alter session set tracefile_identifier = arup;
```

  - Named `<OracleSID>_ora_<spid>_ARUP.trc`

## Analyze the Tracefile

- Oracle provided tool – TKPROF

```
$ tkprof ann1_ora_8420.trc ann1_ora_8420.out
```

- If you want execution plans:

```
$ tkprof ann1_ora_8420.trc ann1_ora_8420.out explain=sh/sh
```

- If you want recursive SQLs

```
$ tkprof ann1_ora_8420.trc ann1_ora_8420.out sys=yes
```

- The insert statements

```
$ tkprof ann1_ora_8420.trc ann1_ora_8420.out insert=tki.sql
```

- All the statements

```
$ tkprof ann1_ora_8420.trc ann1_ora_8420.out record=tkr.sql
```

## tkprof

```
Usage: tkprof tracefile outputfile [explain= ] [table= ]
              [print= ] [insert= ] [sys= ] [sort= ]
  table=schema.tablename   Use 'schema.tablename' with 'explain=' option.
  explain=user/password    Connect to ORACLE and issue EXPLAIN PLAN.
  print=integer    List only the first 'integer' SQL statements.
  aggregate=yes|no
  insert=filename  List SQL statements and data inside INSERT statements.
  sys=no           TKPROF does not list SQL statements run as user SYS.
  record=filename  Record non-recursive statements found in the trace file.
  waits=yes|no     Record summary for any wait events found in the trace file.
  sort=option      Set of zero or more of the following sort options:
    prscnt  number of times parse was called
    prscpu  cpu time parsing
    prsela  elapsed time parsing
    prsdsk  number of disk reads during parse
    prsqry  number of buffers for consistent read during parse
    ...
```

## Extended Tracing

- Activity logging
  - aka 10046 trace
- Enable it by

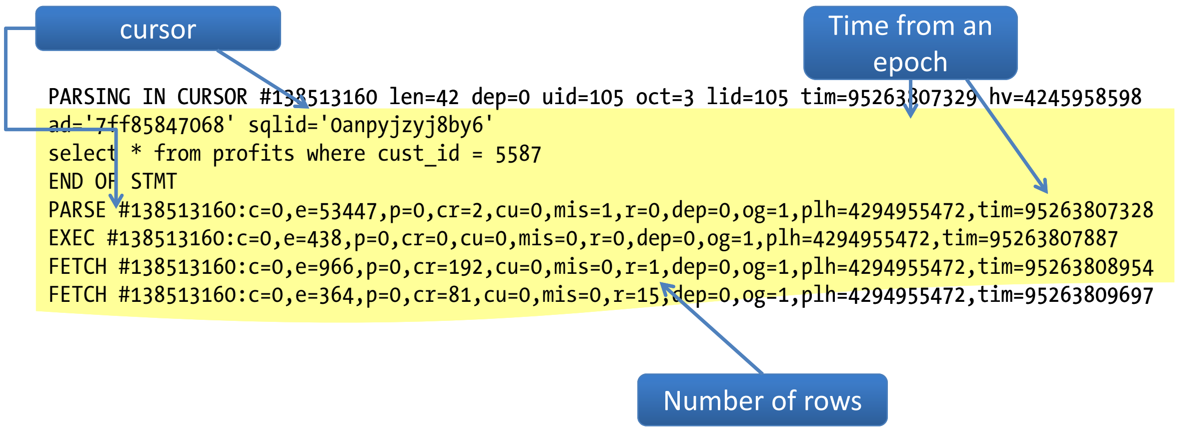
```
alter session set events '10046 trace name context
forever, level 8';
```

- Levels
  - 1 = the regular SQL trace
  - 4 = puts the bind variables
  - 8 = puts the wait information
  - 12 = binds and waits
  - 0 = turns off tracing

## Additional Levels

- Level 16 (11.1+)
  - Level 1 writes exec plan only for the first execution of the cursor
  - This level writes for each execution
- Level 32 (11.1+)
  - Same as level 1 but without the execution plan
- Level 64 (11.2.0.2)
  - If subsequent executions of the cursor takes 1 add'l 60 sec of DB TIME
  - Less overhead since not all exec plan for all execs captured

## Extended Trace Example



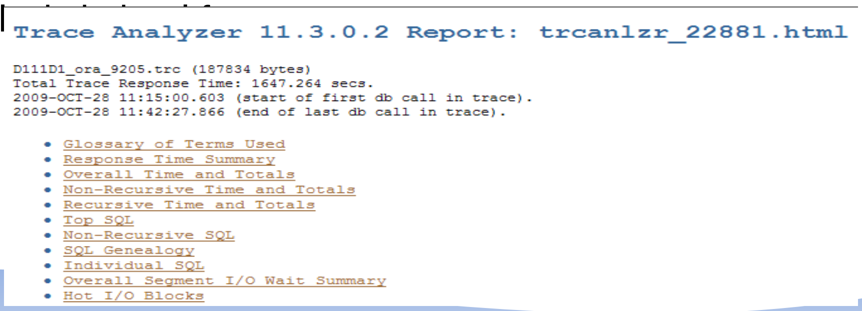
```
PARSING IN CURSOR #138513160 len=42 dep=0 uid=105 oct=3 lid=105 tim=95263807329 hv=4245958598
ad='7ff85847068' sqlid='0anpyjzyj8by6'
select * from profits where cust_id = 5587
END OF STMT
PARSE #138513160:c=0,e=53447,p=0,cr=2,cu=0,mis=1,r=0,dep=0,og=1,plh=4294955472,tim=95263807328
EXEC #138513160:c=0,e=438,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,plh=4294955472,tim=95263807887
FETCH #138513160:c=0,e=966,p=0,cr=192,cu=0,mis=0,r=1,dep=0,og=1,plh=4294955472,tim=95263808954
FETCH #138513160:c=0,e=364,p=0,cr=81,cu=0,mis=0,r=15,dep=0,og=1,plh=4294955472,tim=95263809697
```

## Analyzing Extended Traces

- Limitations of TKProf
  - Extended information nowshown
  - Bind variable values not shown
- Other options
  - Trace Analyzer (Free. From My Oracle Support)
    - Needs database connection
    - Creates a schema, objects
  - Hotsos Profiler (paid)
  - TVD$XTAT (free) http://antognini.ch/downloads/top2/chapter03/tvdxtat_40beta10_20140630.zip
    - No db connection needed
    - Java based; no installation needed

## Trace Analyzer

- It generates
  - The log file of the run. Scan for errors.
  - The tkprof output of the trace file
  - The analysis in text format
  - The anal



Trace Analyzer 11.3.0.2 Report: trcanlzr_22881.html

```
D111D1_ora_9205.trc (187834 bytes)
Total Trace Response Time: 1647.264 secs.
2009-OCT-28 11:15:00.603 (start of first db call in trace).
2009-OCT-28 11:42:27.866 (end of last db call in trace).
```

- Glossary of Terms Used
- Response Time Summary
- Overall Time and Totals
- Non-Recursive Time and Totals
- Recursive Time and Totals
- Top SQL
- Non-Recursive SQL
- SQL Genealogy
- Individual SQL
- Overall Segment I/O Wait Summary
- Hot I/O Blocks

## TVD$XTAT

- Unzip the zip file into a folder/directory
- Onetime config file setup
  - Location of java and the tool
- Analyze the tracefile
  - `C:\> tvdxtat.cmd -i f.trc -o f.html`
- Text format
  - `C:\> tvdxtat.cmd -i f.trc -o f.txt -t text`

## Tracing a Remote Session

- Find out the SID and Serial#
- Option 1

```
dbms_system.set_sql_trace_in_session (sid=>1, serial#=>1,
sql_trace=>true);
```

  - Set sql_trace to FALSE to stop
- Option 2

```
dbms_system.set_ev(si=>1, se=>1, ev=>10046, le=>8, nm=>' ');
```

  - Set le to 0 to stop
- Option 3

```
dbms_support.start_trace_in_session (sid=>1, serial=>1,
waits=>true, binds=>false);
```

The package needs to be created `$OH/rdbms/admin/dbmssupp.sql`

## ORADEBUG

- Oradebug (undocumented)
- Login as SYSDBA
- For the current session

```
SQL> oradebug setmypid;
```

- For a different session. Get the OS PID

```
SQL> oradebug setospid 1;
SQL> oradebug event 10046 trace name context forever, level 8;
```

- To get the current tracefile name

```
SQL> oradebug tracefile_name;
```

- To turn off tracing

```
SQL> oradebug event 10046 trace name context off;
```

## DBMS_MONITOR

- New in 10g

```
begin
  dbms_monitor.session_trace_enable (
    session_id => 1,
    serial_num => 1,
    waits      => true,
    binds      => true
    plan_stat  => 'all_executions');
end;
```

Leave these to trace current session

enasess.sql

- NULL (first_execution default) – level 8
- all_executions – level 16
- never – level 32
- not possible for level 64

- Execute session_trace_disable (…) to disable

## Detecting Tracing

- To find out the sessions where the tracing has been enabled

vsesstrace.sql

```
select sql_trace, sql_trace_waits,
       sql_trace_binds, sql_trace_plan_stats
from v$session
where sid = 255;

SQL_TRAC SQL_T SQL_T SQL_TRACE_
-------- ----- ----- ----------
ENABLED  TRUE  TRUE  ALL EXEC
```

## Individual SQL Statements

- To trace individual SQL Statements
- Get SQL_ID

```
alter session set events 'trace[rdbms.sql_optimizer.*][sql:
0anpyjzyj8by6]';
```

- Run the app
- Disable trace

```
alter session set events 'trace[rdbms.sql_optimizer.*] off';
```

- To get the SQL Trace only

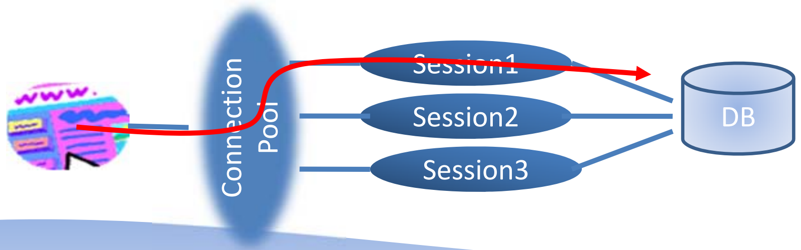
```
alter session set events 'sql_trace[SQL: 0anpyjzyj8by6]';
```

- Turn off

```
alter session set events 'sql_trace off';
```

## The Connection Pool Effect

- Most applications use connection pool
- A "pool" of connections connected to the database
- When the demand on the connection from the pool grows, the pool creates new database sessions
- When the demand lessens, the sessions are disconnected
- The SID is not known



## Tracing in Future Sessions

- Service Names start tracing when any session connected with that service name will be traced

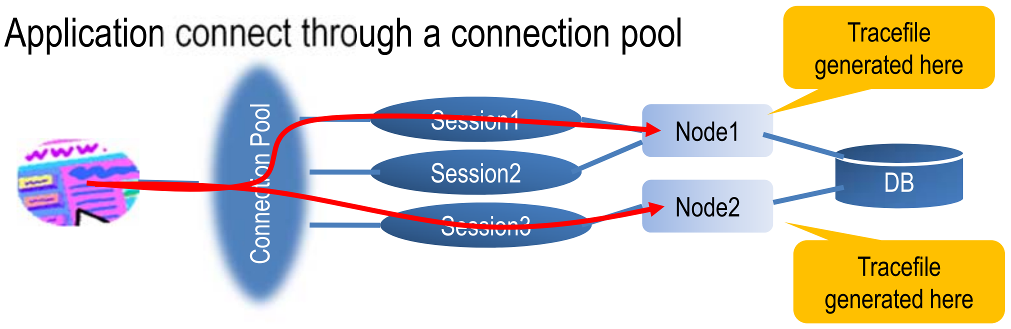
```
begin
   dbms_monitor.serv_mod_act_trace_enable (
        service_name => 'APP',
        action_name  => dbms_monitor.all_actions,
        waits        => true,
        binds        => true
   );
end;
```

Warning: This is case sensitive; so "app" and "APP" are

- This will trace any session connected with service_name APP
- Even future sessions!

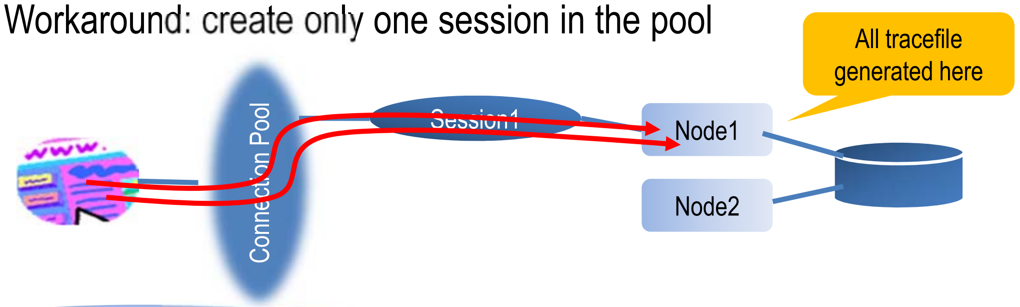
## What's Special About RAC

- Multiple Instances → multiple hosts
- The tracefiles are on different hosts
- Application connect through a connection pool



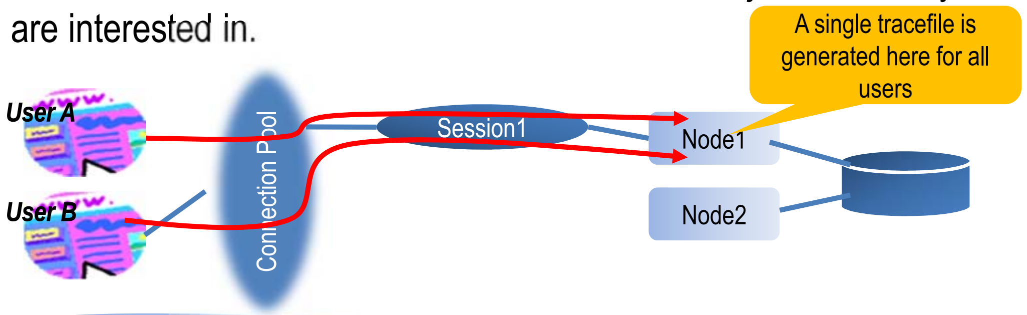
## Multiple Tracefiles

- Tracefiles are generated for each Oracle session
- So, a single user's action can potentially go to many sessions → many tracefiles
- Workaround: create only one session in the pool



## Mixed Activities

- But that does not solve the problem
- The single Oracle session will service activities of many users
- So the tracefile will have activities of all users; not just the user you are interested in.



## Consolidation of Tracefiles

- The trcsess utility comes handy in that case
  - It combines all tracefiles into one!

```
trcsess output=alltraces.trc service=app *.trc
```

  - It creates the tracefile alltraces.trc from all the tracefiles in that directory where activities by all sessions connected with the **app** service
- Now you can treat this new tracefile as a regular tracefile.

```
$ tkprof alltraces.trc alltraces.out sys=no …
```

## Client ID

- Set the Client ID

```
Begin
   dbms_session.set_identifier('CLIENT1');
End;
```

- Check the Client ID

```
select SYS_CONTEXT('userenv', 'client_identifier') from
dual;
```

- For the session

```
select client_identifier from v$session where username =
'SH';
```

## Trace the Client ID Sessions

- Enable

```
dbms_monitor.client_id_trace_enable (
  client_id => ‘CLIENT1',
  waits => true,
  binds => false
 );
```

- Disable

```
dbms_monitor.client_id_trace_disable (
  client_id => ‘CLIENT1'
);
```

## Module and Action

- Set Module

```
dbms_application_info.set_module(
         module_name     => 'MODULE1',
         action_name     => 'ACTION1'
);
```

- Set subsequent actions

```
dbms_application_info.set_action ('ACTION2');
dbms_application_info.set_action ('ACTION3');
```

## Trace Module and Action

- Enable

```
dbms_monitor.serv_mod_act_trace_enable(
 service_name=>'APP',
 module_name=>'MODULE1',
 action_name=>'ACTION1',
 waits=>TRUE, binds=>TRUE
);
```

- Disable

```
dbms_monitor.serv_mod_act_trace_disable(
 service_name=>'APP',
 module_name=>'MODULE1',
 action_name=>'ACTION1');
```

## TRCSESS

- The utility has many options

```
trcsess [output=<output file name >]
  [session=<session ID>] [clientid=<clientid>]
  [service=<service name>] [action=<action name>]
  [module=<module name>] <trace file names>
```

**output**=<output file name> output destination default being standard output.

**session**=<session Id> session to be traced.

Session id is a combination of SID and Serial# e.g. 8.13.

**clientid**=<clientid> clientid to be traced.

**service**=<service name> service to be traced.

## Identifying non-Session Traces

- View DBA_ENABLED_TRACES
  - TRACE_TYPE – scope of tracing: SERVICE, SERVICE_MODULE, SERVICE_MODULE_ACTION
  - PRIMARY_ID: name of that type, e.g. name of the service if SERVICE is enabled
  - QUALIFIER_ID1: module name, if enabled
  - QUALIFIER_ID2: action name, if enabled
  - WAITS: if WAITS are being traced
  - BINDS: if BINDS are being traced
  - PLAN_STATS: all_executions, first_execution or never
  - INSTANCE_NAME: the name of the instance

## Special Cases

- You can enable for the entire database:

  `dbms_monitor.database_trace_enable(…)`
- If tracing the current session and do not have exec privileges for DBMS_MONITOR:

  `dbms_session.session_trace_enable (…)`

## Summary

- Two types of tracing
  - Simple
  - Extended, aka 10046
- Several ways to invoke tracing
- Can start tracing on a different session
- Can set the tracing to trigger if one or more matches:
  - Service, Module, Action, Client_ID
- Can analyze
  - Tkprof
  - Trace Analyzer
  - Other Tools

## *Thank You!*

Blog: arup.blogspot.com
Tweeter: @arupnanda
Facebook.com/ArupKNanda