The *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

```
SQL> 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
    ...
```

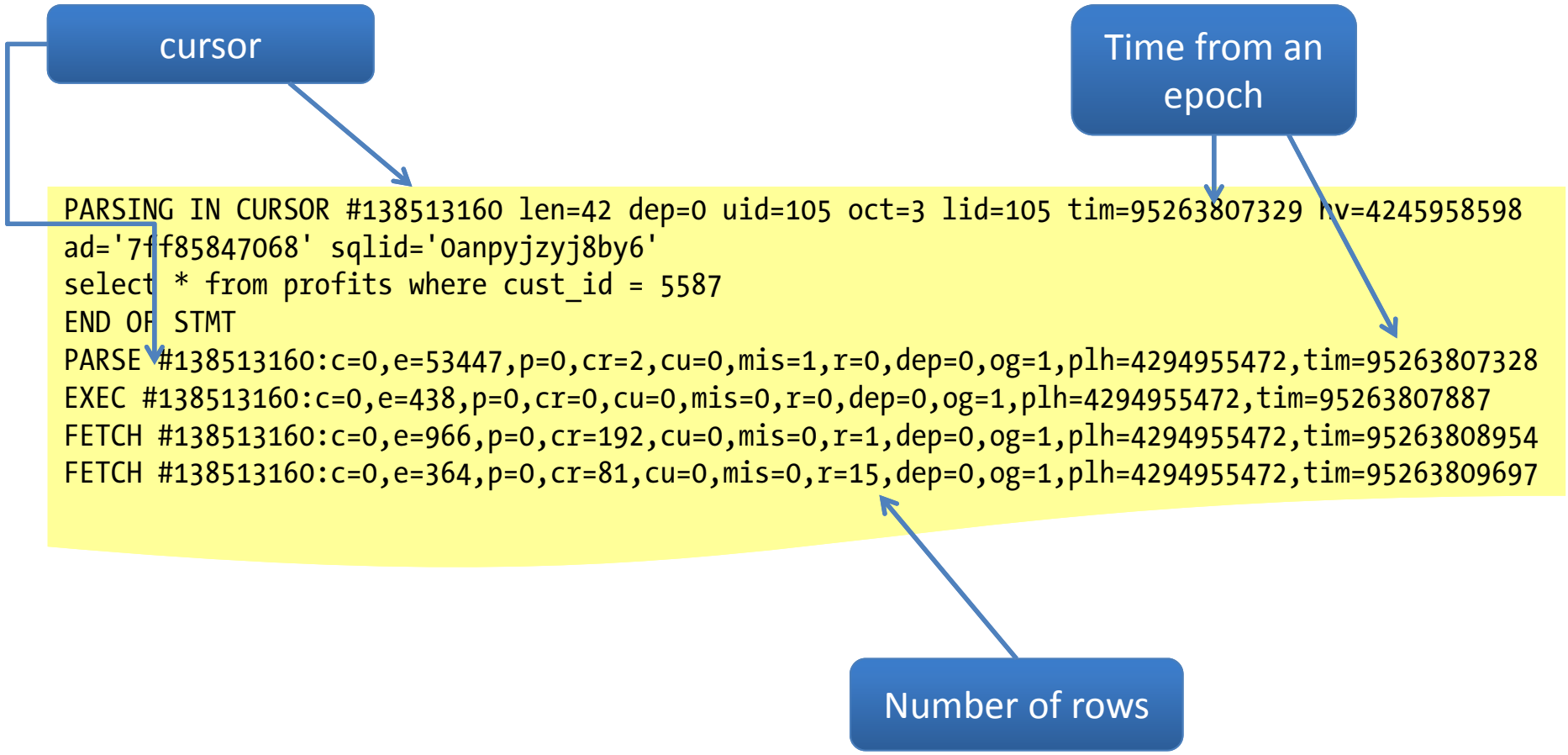
Extended Tracing

- Activity logging
  - aka 10046 trace
- Enable it by

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

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

Extended Trace Example



Analyzing Extended Traces

- Tkprof works too; but no extended information is shown
- Other options
  - Trace Analyzer (Free. From My Oracle Support)
  - Hotsos Profiler (paid)
  - TVD$XTAT (Free. http://antognini.ch/downloads/tvdxtat_40beta9.zip)

Trace Analyzer

- A much better tool to analyze trace files.
- Refer to MetaLink Doc **224270.1** for download and instructions on use
- A small zip file, with bunch of directories
- Connect as SYS and run tacreate.sql to create the Trace Analyzer schema (TRCANLZR)
- Run it

```
cd trca/run
sqlplus trcanlzr/trcanlzr
@trcanlzr <tracefile name in udump dir>
```

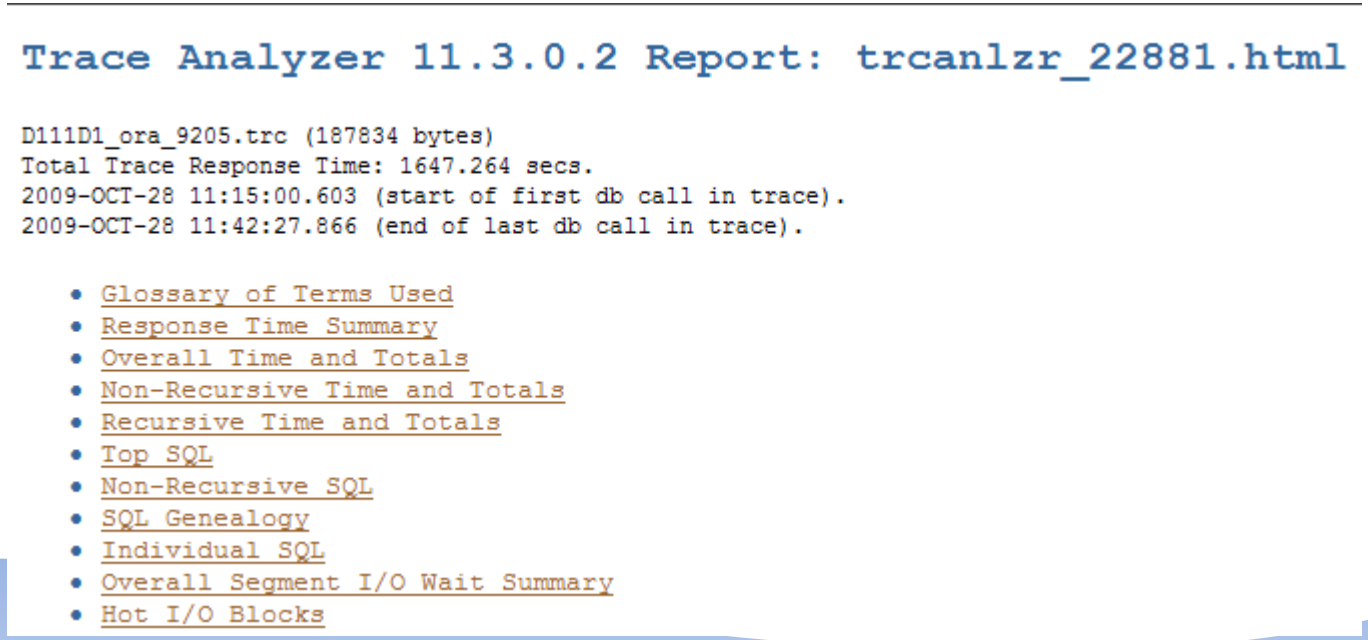
Output

```
Value passed to trcanlzr.sql:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TRACE_FILENAME: D111D1_ora_9205.trc

... analyzing D111D1_ora_9205.trc

Trace Analyzer completed.
Review first trcanlzr_error.log file for possible fatal errors.
Review next trcanlzr_22881.log for parsing messages and totals.

... copying now generated files into local directory

TKPROF: Release 11.1.0.7.0 - Production on Wed Oct 28 11:45:05 2009

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

  adding: trcanlzr_22881_c.html (deflated 90%)
  adding: trcanlzr_22881_c.log (deflated 82%)
  adding: trcanlzr_22881_c.txt (deflated 84%)
  adding: trcanlzr_22881.tkprof (deflated 85%)
  adding: trcanlzr_error.log (deflated 72%)
test of trcanlzr_22881.zip OK

... trcanlzr_22881.zip has been created

TRCANLZR completed.
```

These files are produced in the local directory

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 analysis in html format



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

Enabling Trace in 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

- 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;
```

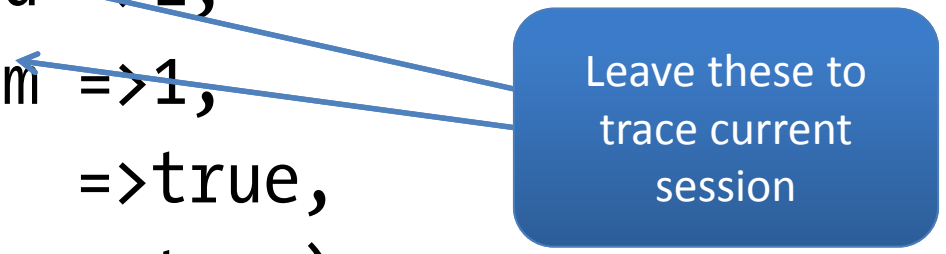
- To turn off tracing

```
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);
end;
```



- Execute `session_trace_disable (…)` to disable

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]';
```

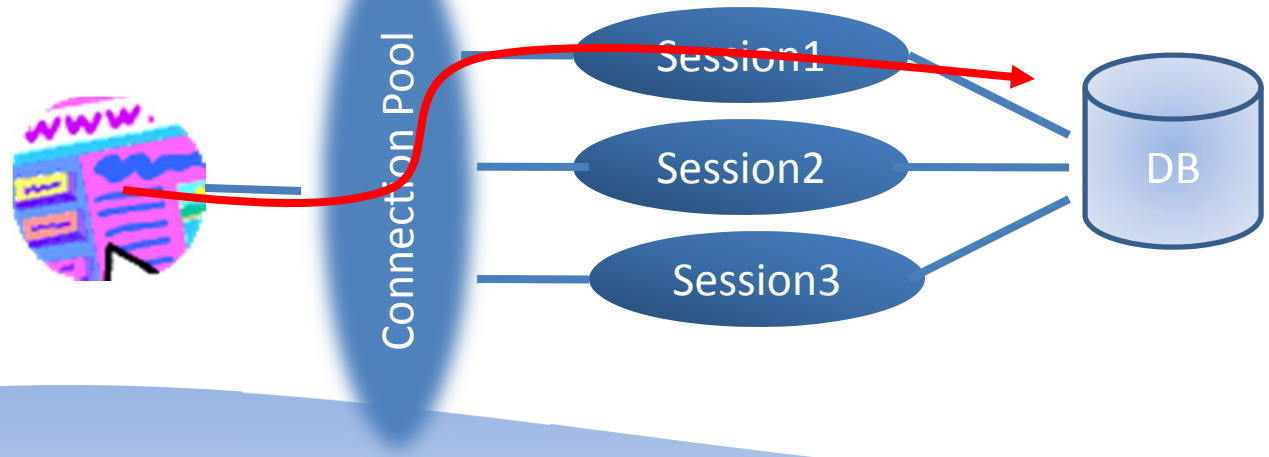
- Turn off

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

CONNECTION POOLS AND RAC

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



Enabling Tracing in Future Sessions

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

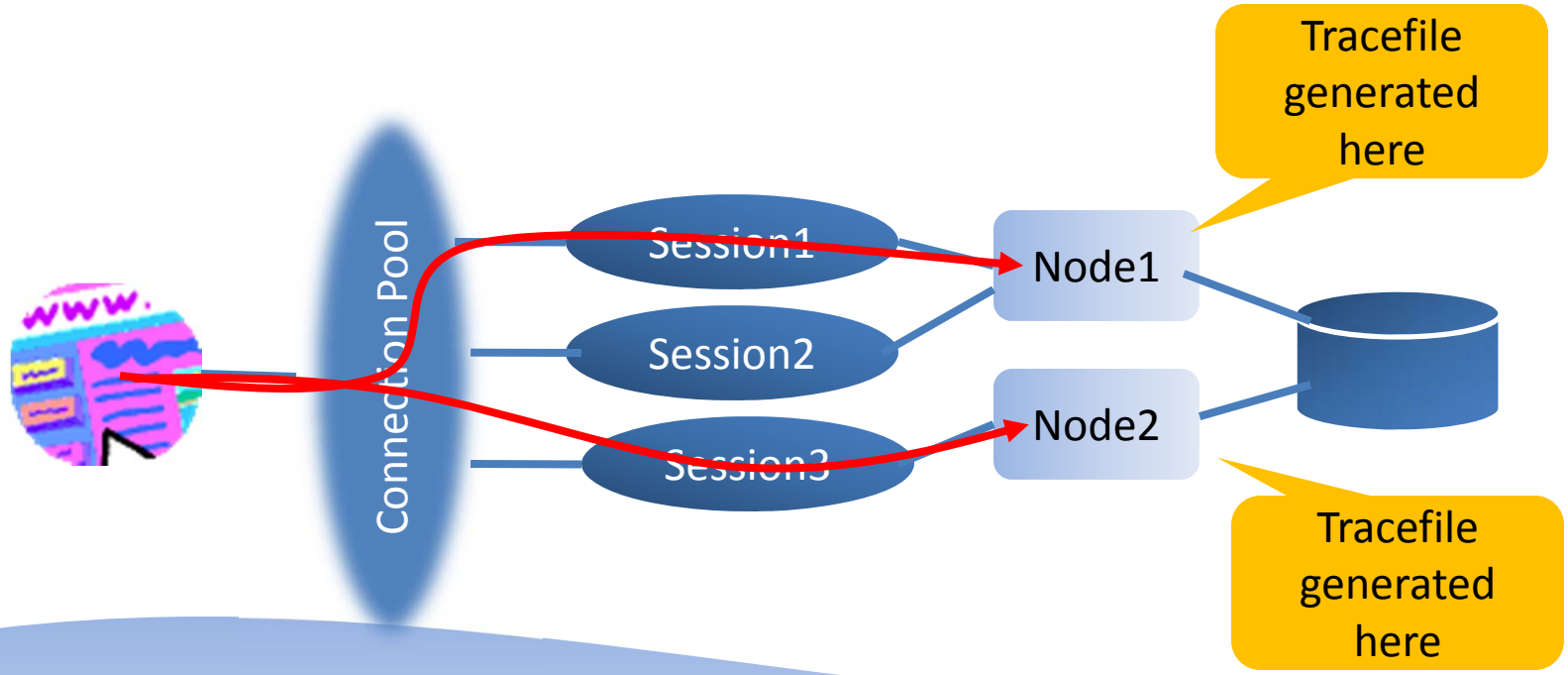
```
begin
    dbms_monitor.serv_mod_act_trace_enable (
         service_name => 'APP',
         action_name  => dbms_monitor.all_actions,
         waits        => true,
         binds        => true
    );
end;
```

Warning: This is case sensitive; so “app” and “APP” are different.

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

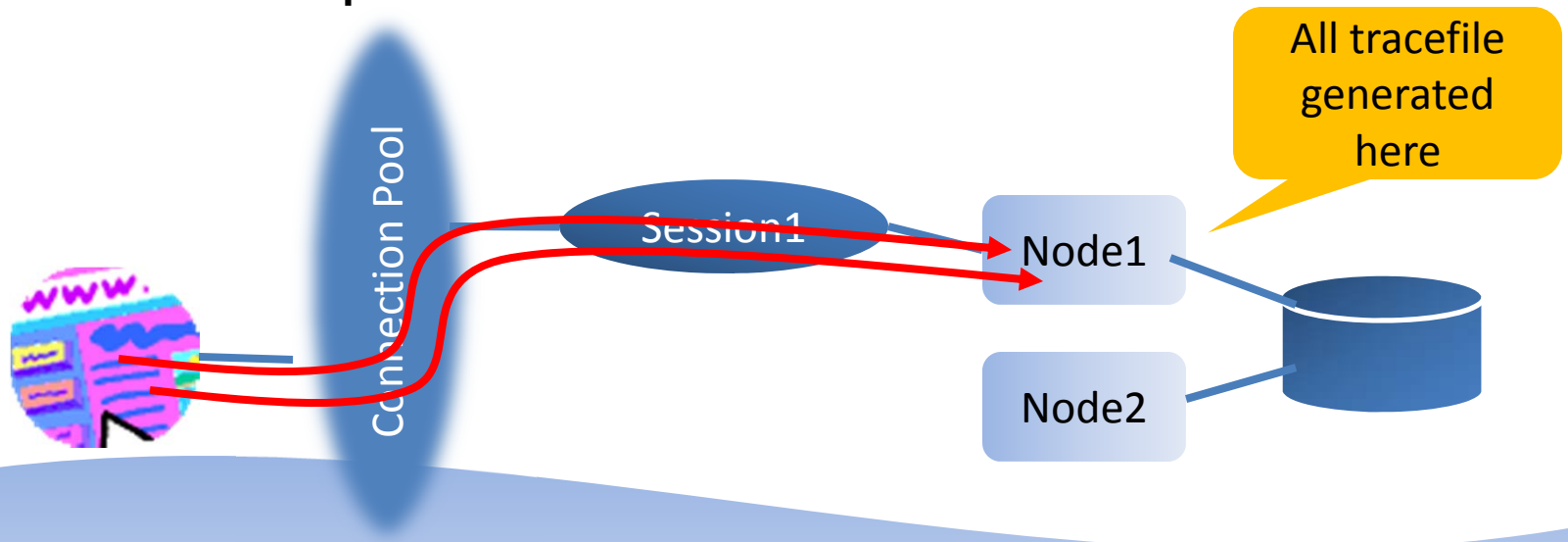
What's Special About RAC

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



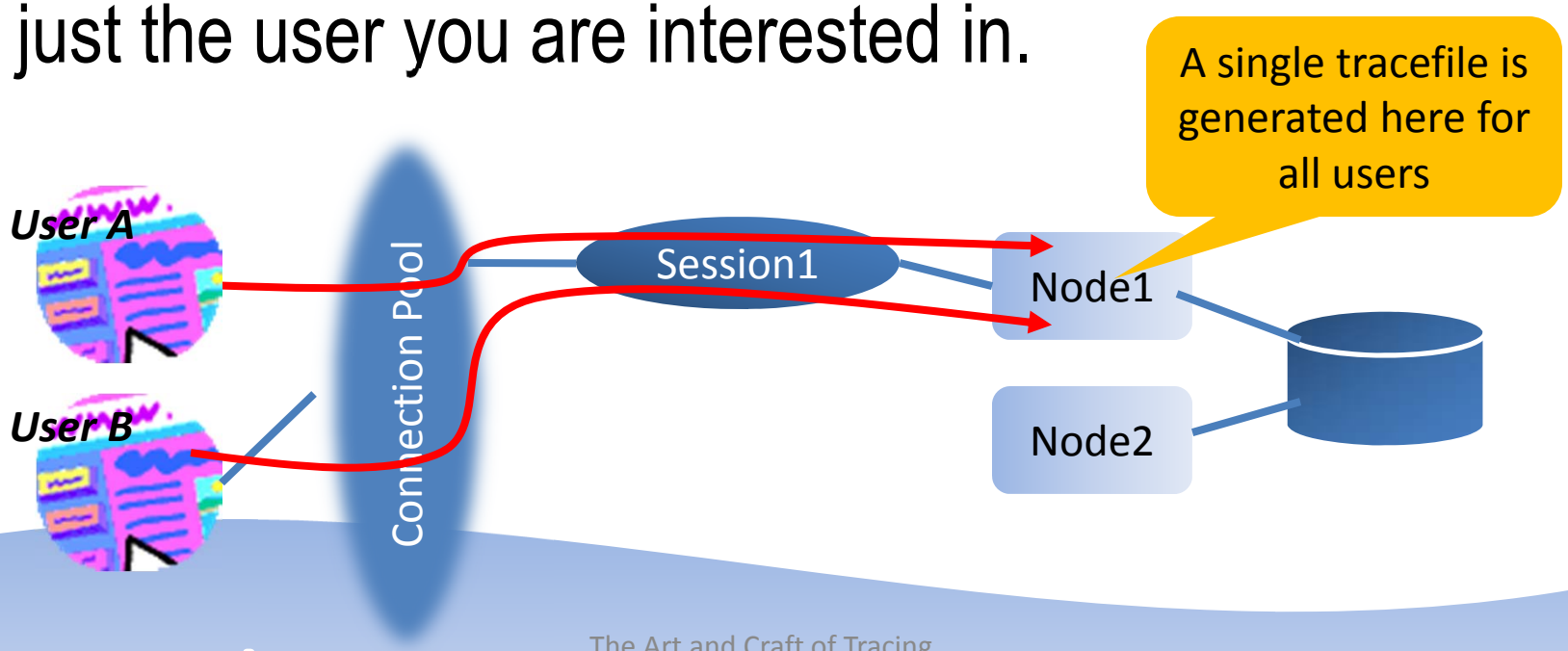
Multiple Tracefiles

- 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 connection 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 …
```

DIFFERENTIATING AMONG APPS

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.

action=<action name> action to be traced.

module=<module name> module to be traced.

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
- Can analyze
  - Tkprof
  - Trace Analyzer
  - Other Tools

Thank You!

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