Advanced Oracle Troubleshooting

No magic is needed,
systematic approach will do

Tanel Põder

http://www.tanelpoder.com

ORACLE | CERTIFIED MASTER

Introduction

- About me:
  - Occupation: DBA, researcher, consultant
  - Expertise: Oracle internals geek, *End-to-end* performance & scalability, troubleshooting
  - Oracle experience: 12+ years as DBA
  - Certification: OCM (2002) OCP (1999)
  - Professional affiliations: OakTable Network
  - Blog: http://blog.tanelpoder.com
  - Seminar: http://blog.tanelpoder.com/seminar/

ORACLE | CERTIFIED MASTER

Introduction

- About this presentation:
  - Systematic *approach*, rather than *methodology*
  - Use *right* tools for *right* problems
  - Break complex problems down to simple problems
  - Therefore, use simple tools for simple problems
  - In other words, use a *systematic approach* and life will be easier!
- Less slides, more action!
- All scripts used here are freely available:
  - http://www.tanelpoder.com

Metalink forum thread example:

Query is taking long time to complete--Need urgent help
From: …… 05-Jun-08 08:40
Subject: Re : Query is taking long time to complete--Need urgent help

Posting TKPROF may help.

From: …… 05-Jun-08 21:33
Subject: Re : Query is taking long time to complete--Need urgent help

Just a hunch, but try changing optimizer mode to FIRST_ROWS.

From: …… 06-Jun-08 03:08
Subject: Re : Query is taking long time to complete--Need urgent help

Hi,
Please try using the HINTS like FIRST_ROWS or USE_HASH. This may solve the problem.

Simple (but common) question:

What the $#*&%! is that session doing?

Non-systematic troubleshooting

- Check alert.log…
- Check for disk and tablespace free space…
- Check for locks…
- Check for xyz…

"We did a healthcheck and everything looks OK!"

?????!

Semi-systematic troubleshooting

- Quick check for *usual suspects*
  - System load, locks, etc…
- Look into Statspack (or AWR)…
- Enable sql_trace…

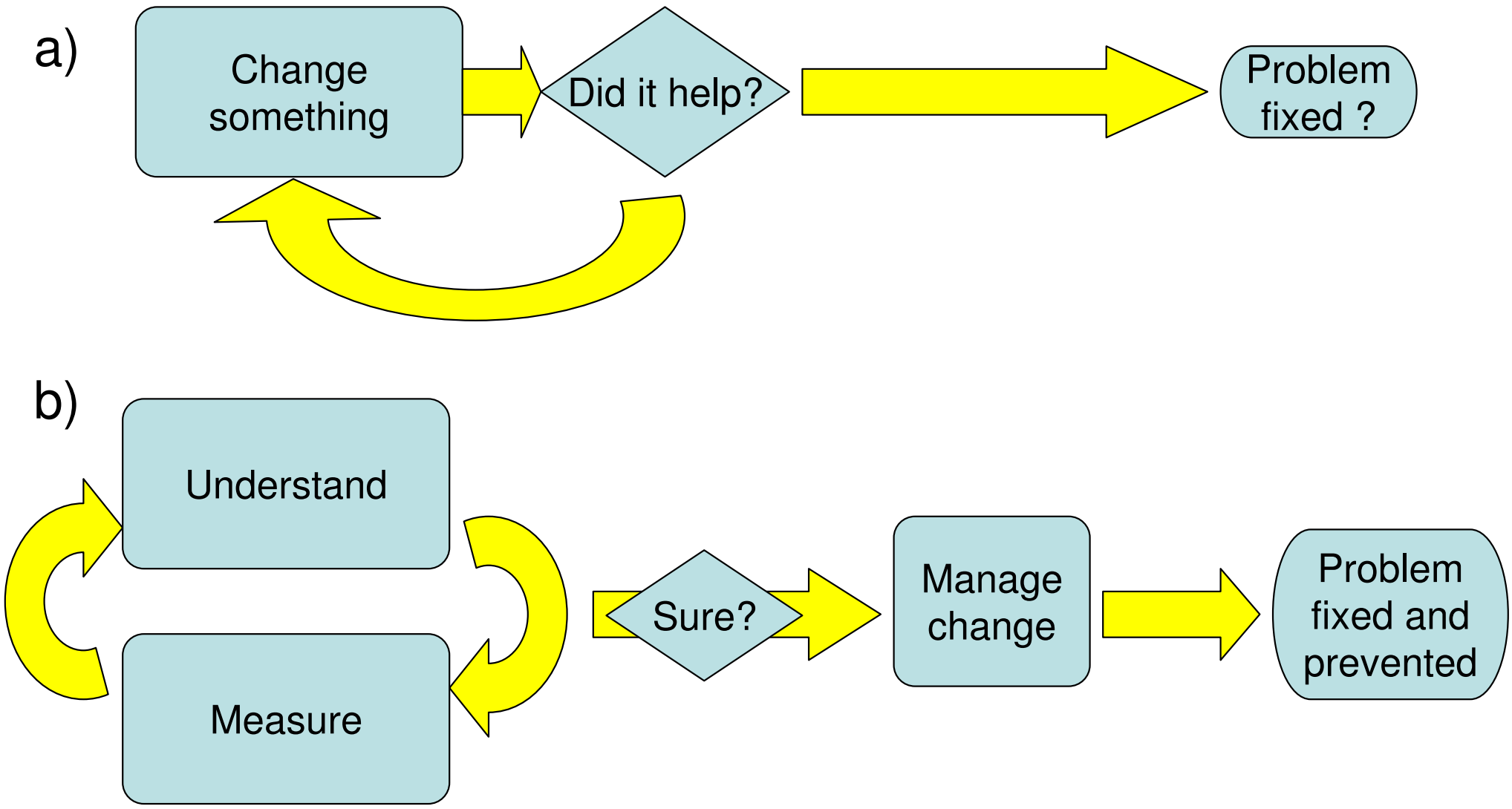
…then what?

Systematic troubleshooting

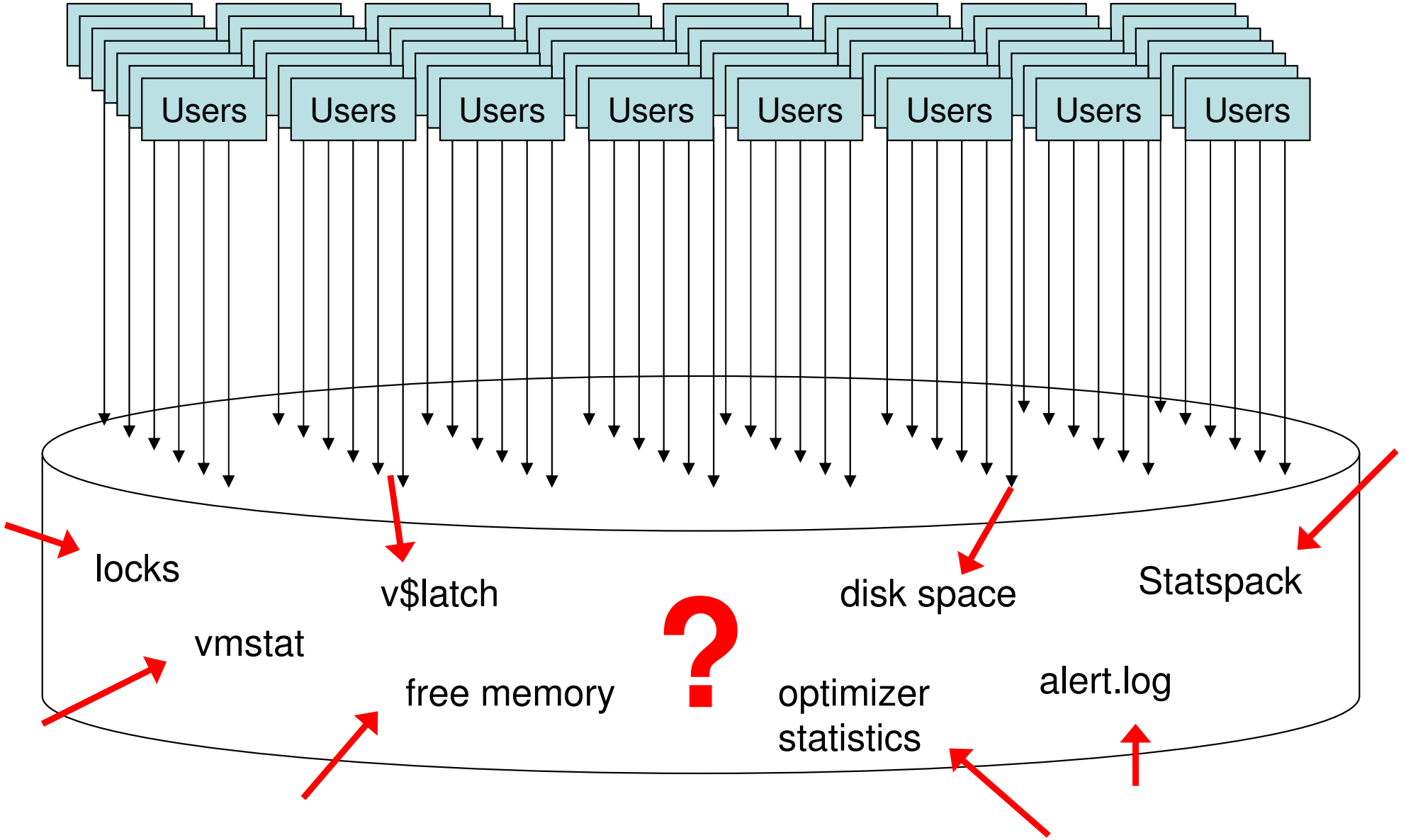
Demo

Troubleshooting approaches

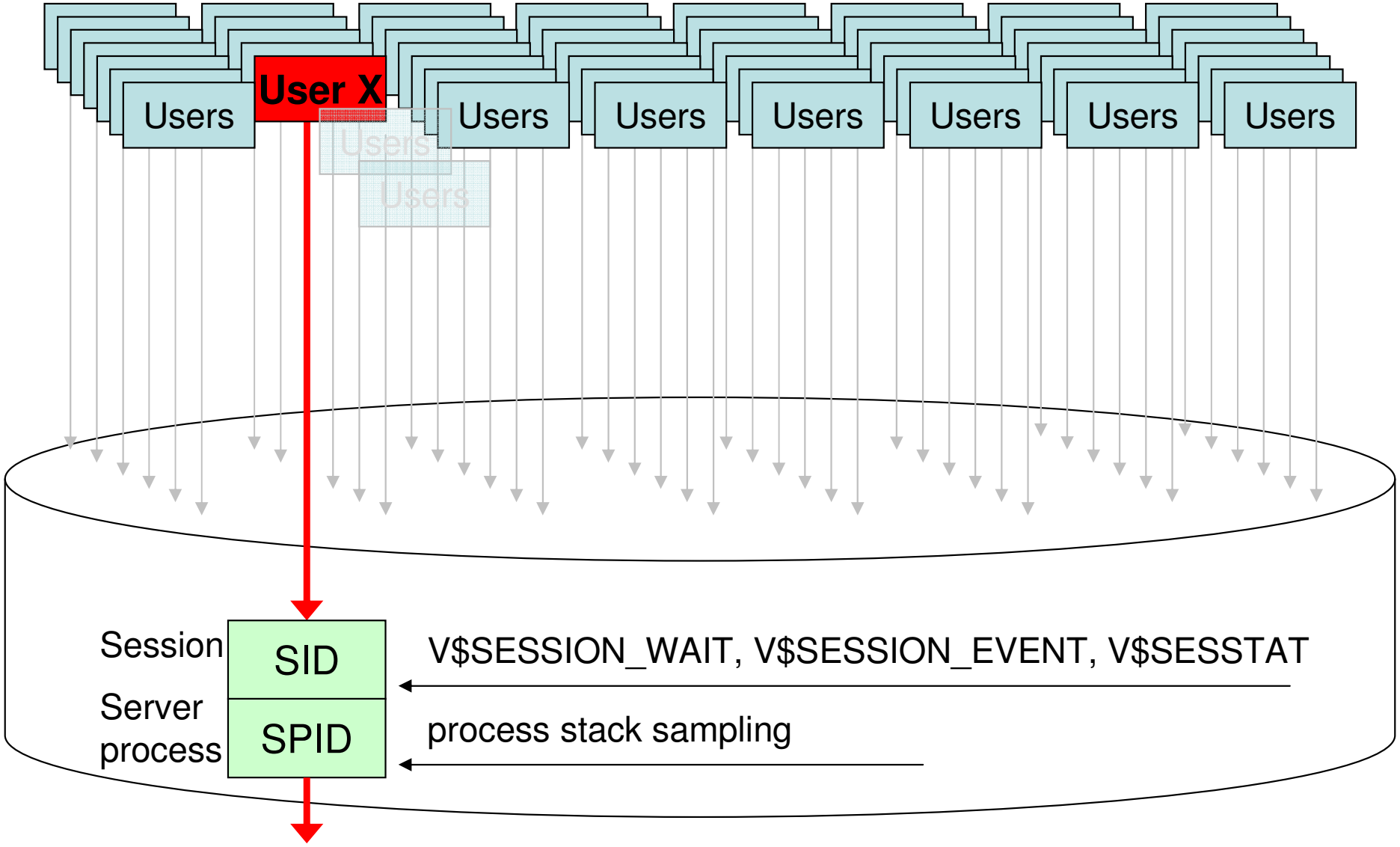
- How do you solve problems?



Non-Systematic approach!

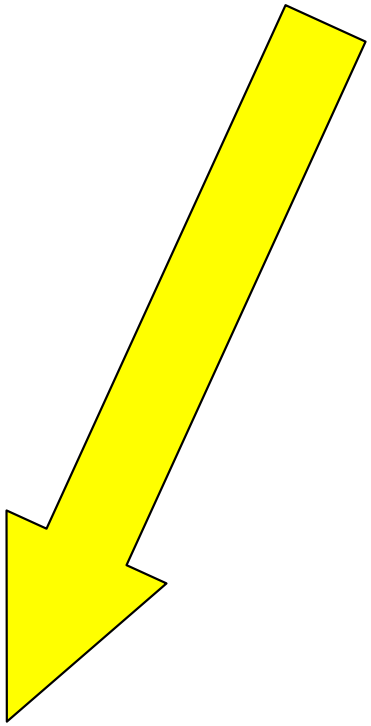


Systematic approach!



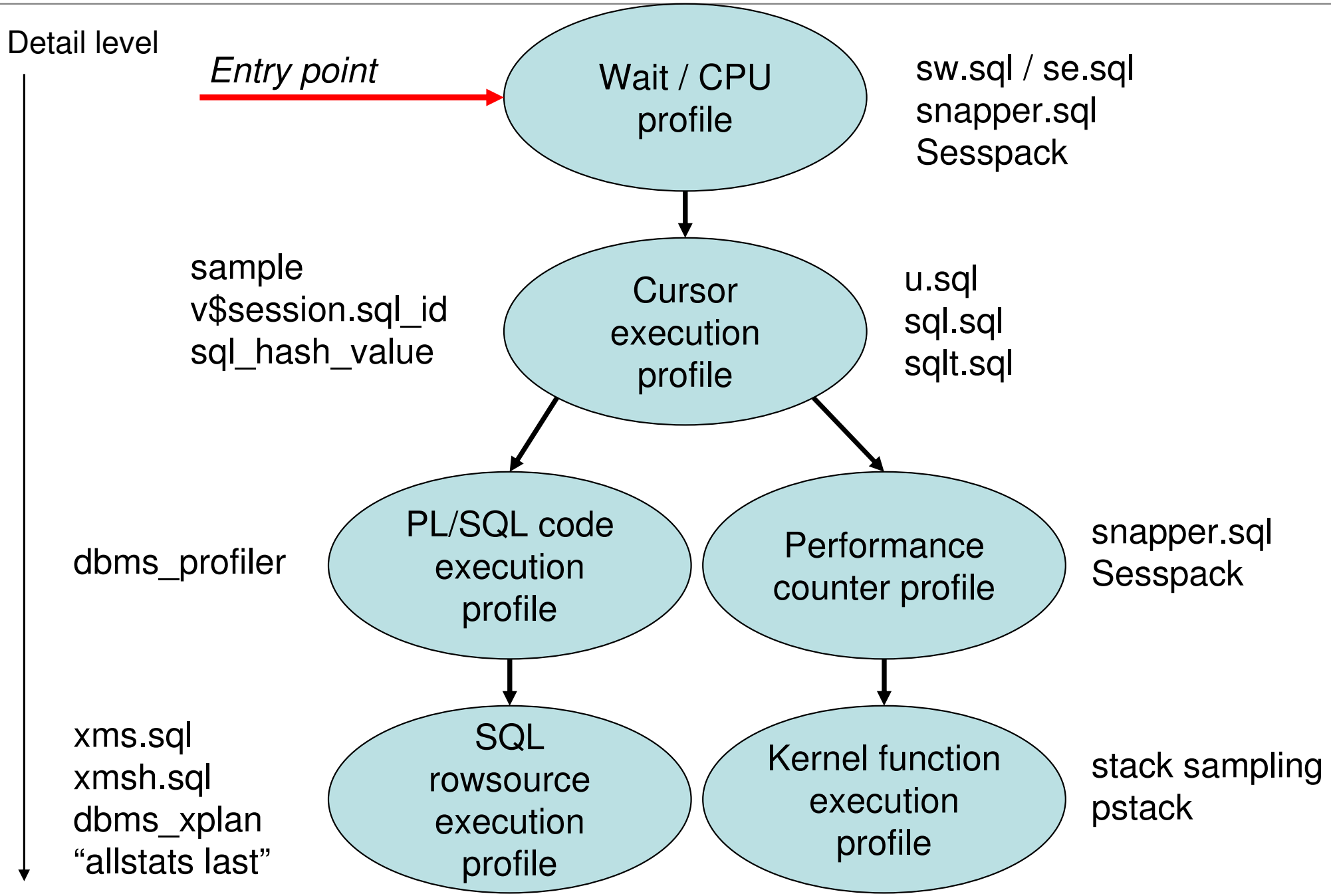
Systematic troubleshooting

- Understand the "flow" of a server process
- …and how to measure it
- …then measure it
- …step by step
- …using right tool at right step



- ...fix the problem *once you understand it*

Right tools for *measuring* right problems



sw.sql and Snapper demo

```
SQL> @sw 114

    SID STATE   EVENT                                     SEQ# SEC_IN_WAIT         P1         P2         P3 P1TRANSL
------- ------- ---------------------------------------- ---------- ----------- ---------- ---------- ---------- ----------------------
    114 WAITING enq: TX - row lock contention                21           9 1415053318     131081       2381 0x54580006: TX mode 6

SQL> @sw &mysid

    SID STATE   EVENT                                     SEQ# SEC_IN_WAIT         P1         P2         P3 P1TRANSL
------- ------- ---------------------------------------- ---------- ----------- ---------- ---------- ---------- ----------------------
    107 WORKING On CPU / runqueue                            89           0 1413697536          1          0

SQL>
SQL> @sn 5 &mysid

-- Session Snapper v1.06 by Tanel Poder ( http://www.tanelpoder.com )

-------------------------------------------------------------------------------------------------------------------------------------
HEAD,     SID, SNAPSHOT START   ,  SECONDS, TYPE, STATISTIC                               ,         DELTA,  DELTA/SEC,     HDELTA, HDELTA/SEC
-------------------------------------------------------------------------------------------------------------------------------------
DATA,       9, 20080221 22:05:08,        5, STAT, recursive calls                       ,             1,          0,          1,         .2
DATA,       9, 20080221 22:05:08,        5, STAT, recursive cpu usage                   ,             1,          0,          1,         .2
DATA,       9, 20080221 22:05:08,        5, STAT, session pga memory max                ,         25292,       5058,     25.29k,      5.06k
DATA,       9, 20080221 22:05:08,        5, STAT, calls to get snapshot scn: kcmgss     ,             1,          0,          1,         .2
DATA,       9, 20080221 22:05:08,        5, STAT, workarea executions - optimal         ,            18,          4,         18,        3.6
DATA,       9, 20080221 22:05:08,        5, STAT, execute count                         ,             1,          0,          1,         .2
DATA,       9, 20080221 22:05:08,        5, STAT, sorts (memory)                        ,            11,          2,         11,        2.2
DATA,       9, 20080221 22:05:08,        5, STAT, sorts (rows)                          ,          1904,        381,       1.9k,      380.8
DATA,       9, 20080221 22:05:08,        5, WAIT, PL/SQL lock timer                     ,       4999649,     999930,         5s,   999.93ms
--  End of snap 1

PL/SQL procedure successfully completed.
```

If this doesn't help…

…Where to look next?

Session troubleshooting sequence

1. Oracle Wait Interface - *response **TIME!***

?

2. v$sesstat performance counters - *hints, indicators*

?

3. Process stack - *truth about process execution*

!!!

Process stack demos

```
$ pstack 5855
#0  0x00c29402 in __kernel_vsyscall ()
#1  0x005509e4 in semtimedop () from /lib/libc.so.6
#2  0x0e5769b7 in sskgpwwait ()
#3  0x0e575946 in skgpwwait ()
#4  0x0e2c3adc in ksliwat ()
#5  0x0e2c3449 in kslwaitctx. ()
#6  0x0b007261 in kjusuc ()
#7  0x0c8a7961 in ksipgetctx ()
#8  0x0e2d4dec in ksqcmi ()
#9  0x0e2ce9b8 in ksqgtlctx ()
#10 0x0e2cd214 in ksqgelctx. ()
#11 0x08754afa in ktcwit1 ()
#12 0x0e39b2a8 in kdddgb ()
#13 0x08930c80 in kdddel ()
#14 0x0892af0f in kaudel ()
#15 0x08c3d21a in delrow ()
#16 0x08e6ce16 in qerdlFetch ()
#17 0x08c403c5 in delexe ()
#18 0x0e3c3fa9 in opiexe ()
#19 0x08b54500 in kpoal8 ()
#20 0x0e3be673 in opiodr ()
#21 0x0e53628a in ttcpip ()
#22 0x089a87ab in opitsk ()
#23 0x089aaa00 in opiino ()
#24 0x0e3be673 in opiodr ()
#25 0x089a4e76 in opidrv ()
#26 0x08c1626f in sou2o ()
#27 0x08539aeb in opimai_real ()
#28 0x08c19a42 in ssthrdmain ()
#29 0x08539a68 in main ()
```

Where to look up the meaning of Oracle kernel function names?

1) Metalink:

175982.1 ORA-600 Lookup Error Categories

453521.1 ORA-04031 “KSFQ Buffers” ksmlgpalloc

Search: <function> "executable entry point”

2) Oracle views

v$latch_misses (lm.sql)

v$latchholder (latchprofx.sql)

v$fixed_view_definition (d.sql, f.sql)

3) Internet search

Getting stack traces

- OS stack dumper
  - pstack - Solaris, Linux, HP-UX
  - procstack - AIX
  - gdb bt, mdb $c
  - Procwatcher (Metalink note: 459694.1)
- Windows
  - windbg, procexp - but no symbolic function names in oracle.exe :(
- Oracle internal
  - oradebug short_stack
  - oradebug dump errorstack 1
  - alter session set events '942 trace name errorstack'

Troubleshooting SQL plan execution

...first requires *understanding* how SQL is executed!

What is an execution plan?

- For Oracle server:
  - *Parsed*, optimized and compiled SQL code kept inside library cache
- For DBAs and developers:
  - Text or graphical representation of SQL execution flow
- Often known as *explain plan*
  - To be correct in terms, explain plan is just a tool, command in Oracle
  - Explain plan outputs textual representation of execution plan into plan table
  - DBAs/developers report human readable output from plan table

One slide for getting execution plan

- Starting from 9.2 the recommended way is:
  - explain plan for <statement>
  - select * from table(dbms_xplan.display)
- In 10g
  - the *autotrace* also uses dbms_xplan
  - set autotrace on
  - or select * from table(dbms_xplan.display_cursor())
- In 11g
  - DBMS_SQLTUNE.REPORT_SQL_MONITOR
- Other methods
  - sql_trace / event 10046 trace + tkprof utility
  - v$sql_plan
  - setting event 10132 at level 1
  - 3rd party tools (which use explain plan anyway)

Parse stages

- Syntactic check
  - Syntax, keywords, sanity
- Semantic check
  - Whether objects referenced exist, are accessible (by permissions) and are usable

soft parse

- View merging
  - Queries are written to reference base tables
  - Can merge both stored views and inline views
- Query transformation
  - Transitivity, etc (example: if a=1 and a=b then b=1)
- Optimization
- Query execution plan (QEP) generation
- Loading SQL and execution plan in library cache

hard parse

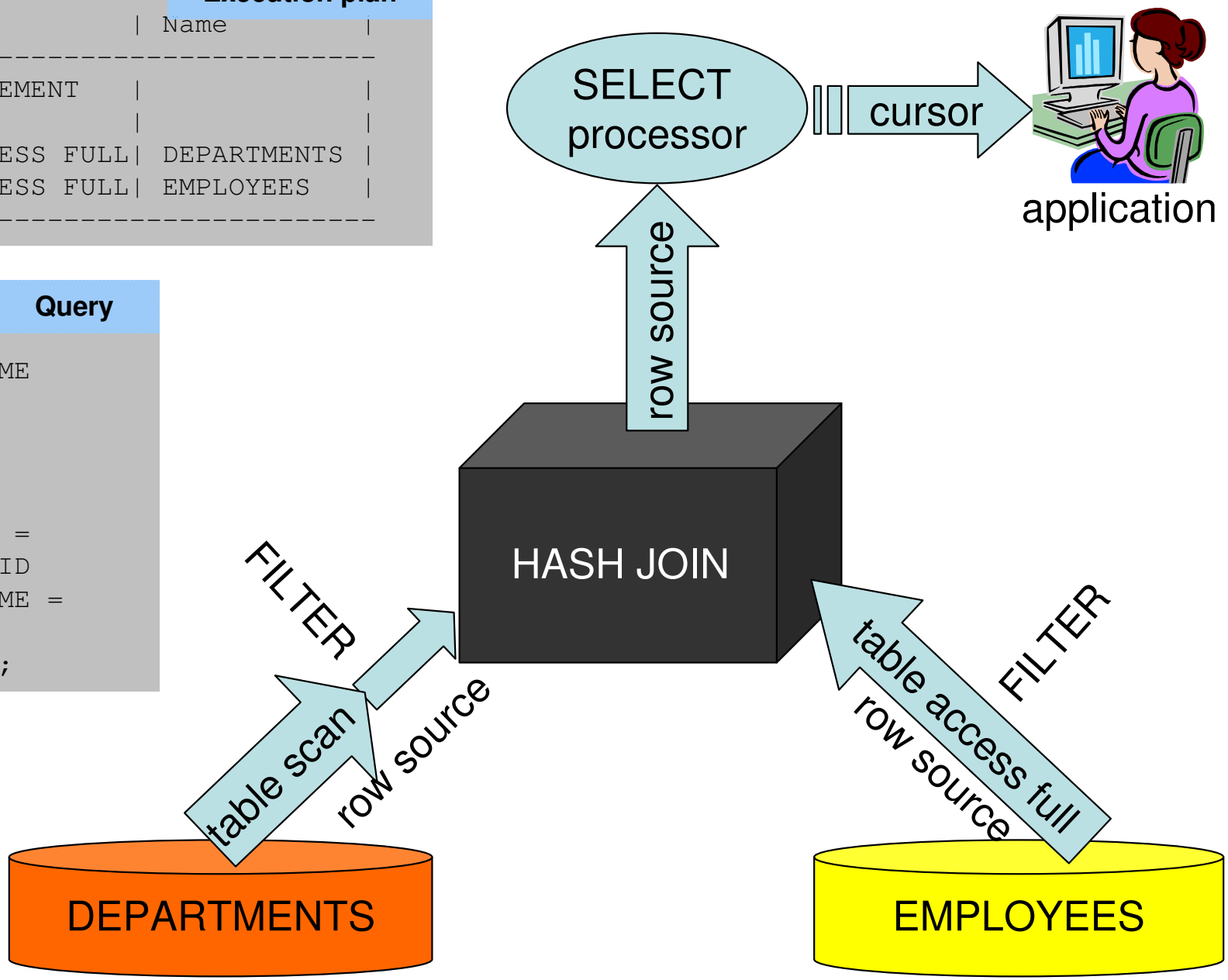
SQL execution data flow - basics

Execution plan

```
-----------------------------------------
| Id  | Operation           | Name      |
-----------------------------------------
|   0 | SELECT STATEMENT    |           |
|*  1 |  HASH JOIN          |           |
|*  2 |   TABLE ACCESS FULL| DEPARTMENTS |
|*  3 |   TABLE ACCESS FULL| EMPLOYEES   |
-----------------------------------------
```

Query

```
SELECT
    E.LAST_NAME,
    D.DEPARTMENT_NAME
FROM
    EMPLOYEES E,
    DEPARTMENTS D
WHERE
    E.DEPARTMENT_ID =
      D.DEPARTMENT_ID
AND D.DEPARTMENT_NAME =
      'Sales'
AND E.SALARY > 2000;
```

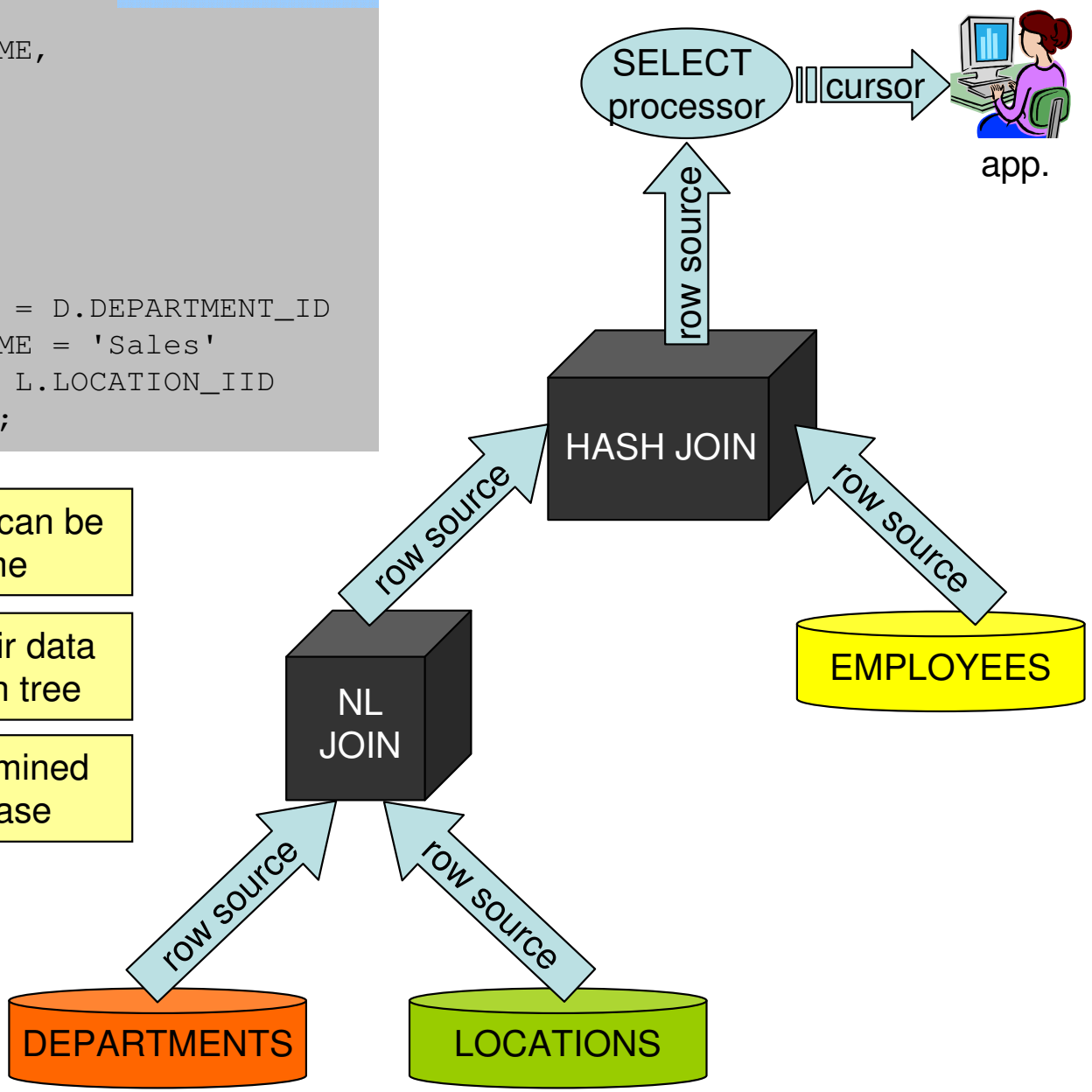


SQL execution data flow - multitable joins

Multiple joins

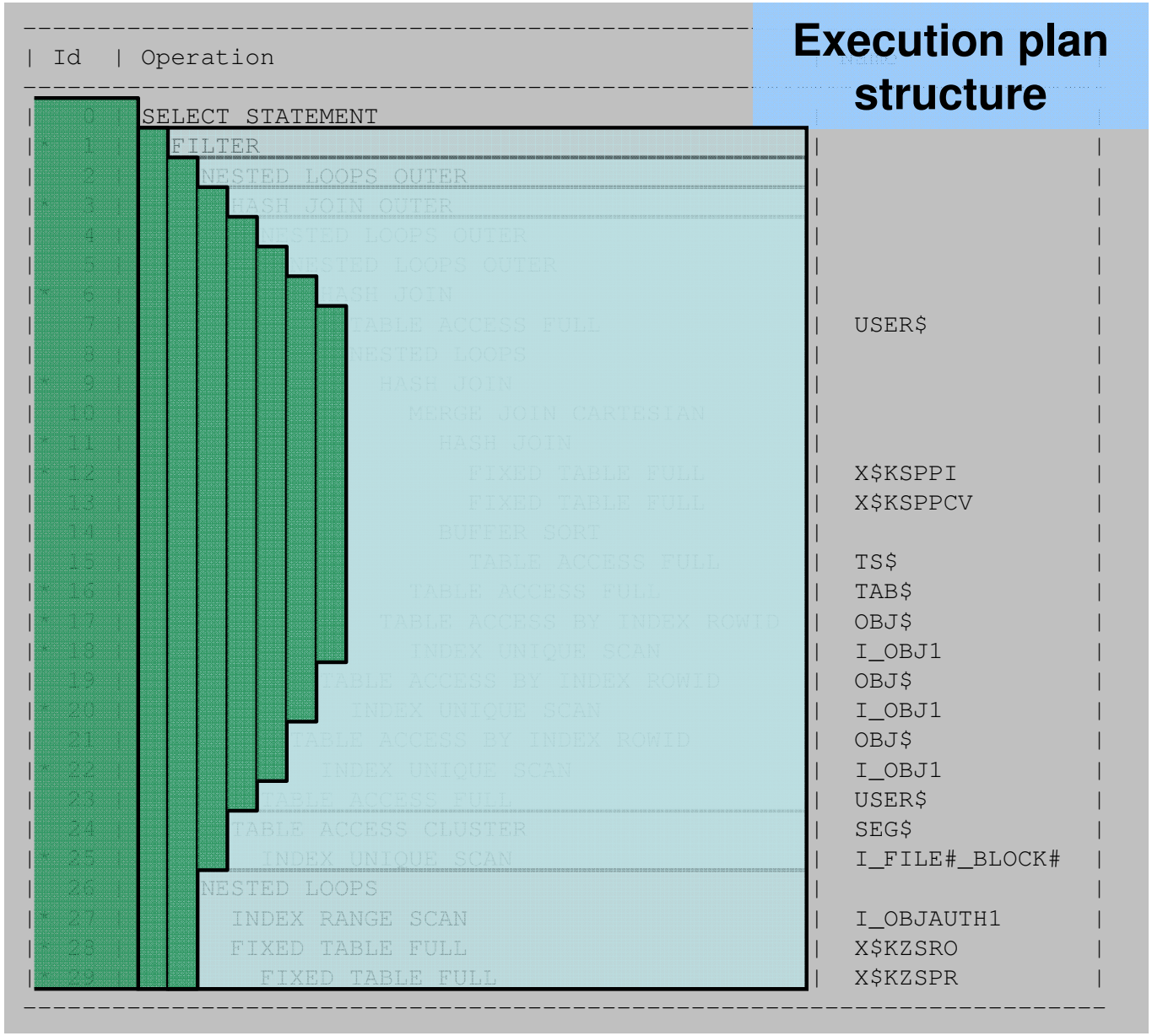
```
SELECT
    E.LAST_NAME,
    D.DEPARTMENT_NAME,
    L.CITY
FROM
    EMPLOYEES E,
    DEPARTMENTS D,
    LOCATIONS L
WHERE
    E.DEPARTMENT_ID = D.DEPARTMENT_ID
AND D.DEPARTMENT_NAME = 'Sales'
AND D.LOCATION_ID = L.LOCATION_IID
AND E.SALARY > 2000;
```

- Only two row sources can be joined together at a time
- Row sources pass their data "up" the execution plan tree
- The join order is determined during optimization phase



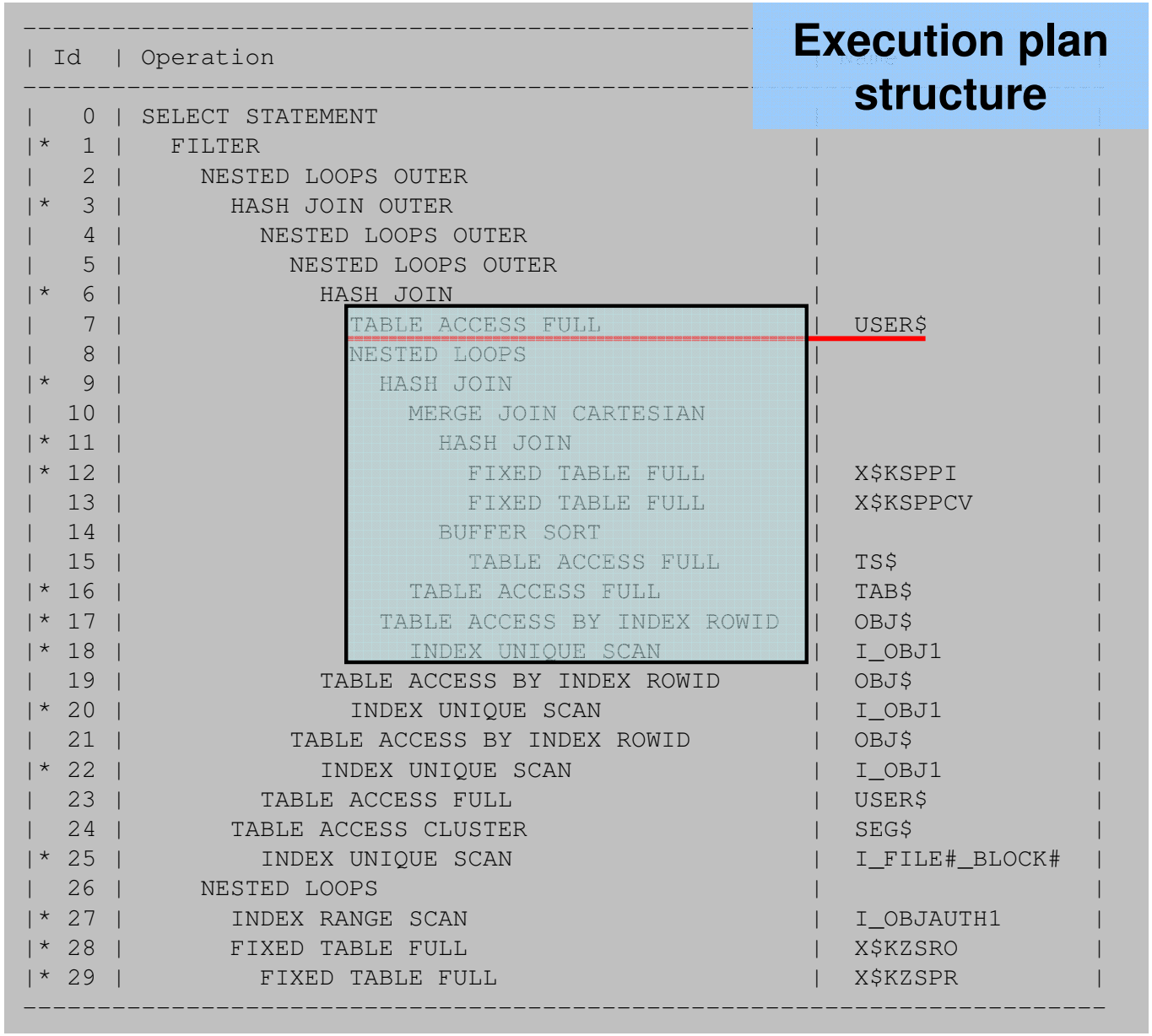
Reading execution plans: Rule 1

- *Parent operations get input only from their children*



Reading execution plans: Rule 2

- *Data access starts from the first line without children*



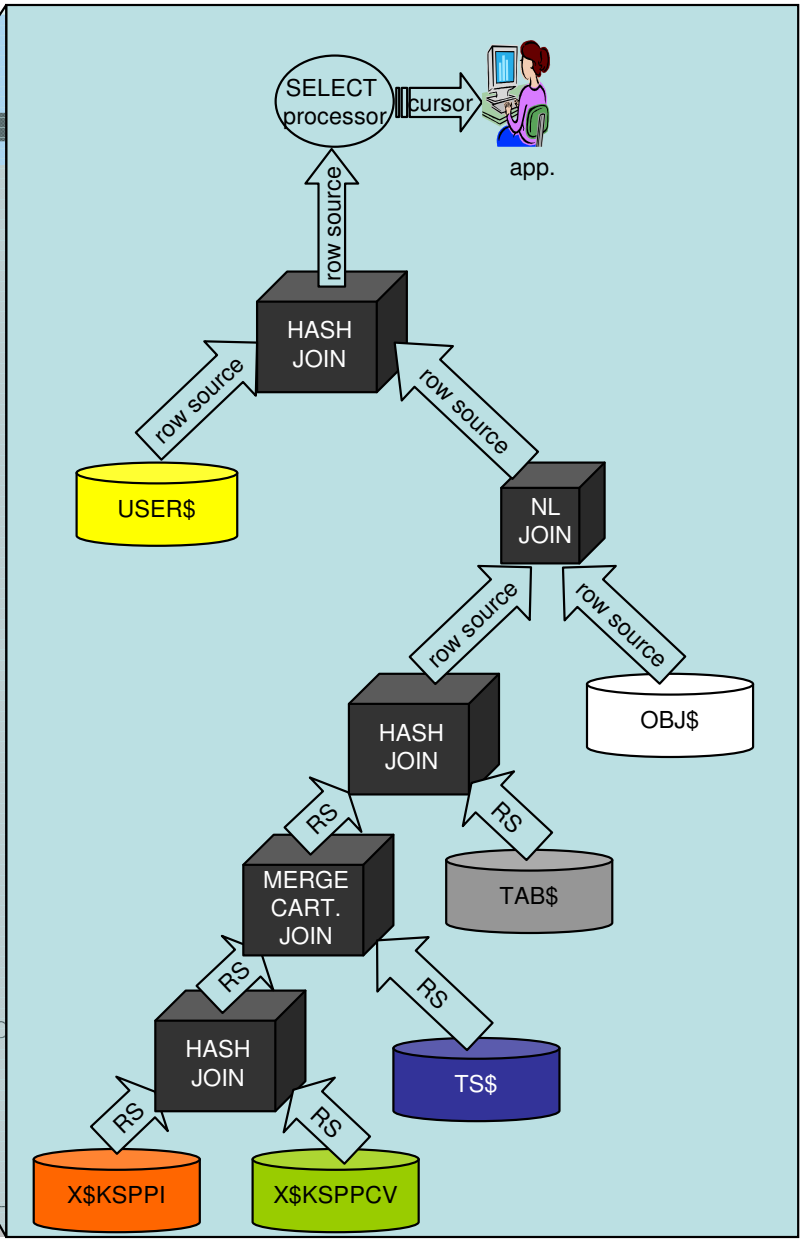
First operation with no children (leaf operation) **accesses** data

Cascading rowsources

- *Rows "cascade" upwards to parent rowsources from children*

```
-----------------------------------------------------
| Id  | Operation                                     |
-----------------------------------------------------
|   0 | SELECT STATEMENT                              |
|*  1 |   FILTER                                      |
|   2 |     NESTED LOOPS OUTER                        |
|*  3 |       HASH JOIN OUTER                         |
|   4 |         NESTED LOOPS OUTER                    |
|   5 |           NESTED LOOPS OUTER                  |
|*  6 |             HASH JOIN                         |
|   7 |               TABLE ACCESS FULL               |  USER$
|   8 |               NESTED LOOPS                    |
|*  9 |                 HASH JOIN                     |
|  10 |                   MERGE JOIN CARTESIAN        |
|* 11 |                     HASH JOIN                 |
|* 12 |                       FIXED TABLE FULL        |  X$KSPPI
|  13 |                       FIXED TABLE FULL        |  X$KSPPCV
|  14 |                     BUFFER SORT               |
|  15 |                       TABLE ACCESS FULL       |  TS$
|* 16 |                   TABLE ACCESS FULL           |  TAB$
|* 17 |                 TABLE ACCESS BY INDEX ROWID   |  OBJ$
|* 18 |                   INDEX UNIQUE SCAN           |  I_OBJ1
|  19 |               TABLE ACCESS BY INDEX ROWID     |  OBJ$
|* 20 |                 INDEX UNIQUE SCAN             |  I_OBJ1
|  21 |             TABLE ACCESS BY INDEX ROWID       |  OBJ$
|* 22 |               INDEX UNIQUE SCAN               |  I_OBJ1
|  23 |           TABLE ACCESS FULL                   |  USER$
|  24 |         TABLE ACCESS CLUSTER                  |  SEG$
|* 25 |           INDEX UNIQUE SCAN                   |  I_FILE#_BLO
|  26 |       NESTED LOOPS                            |
|* 27 |         INDEX RANGE SCAN                      |  I_OBJAUTH1
|* 28 |         FIXED TABLE FULL                      |  X$KZSRO
|* 29 |           FIXED TABLE FULL                    |  X$KZSPR
-----------------------------------------------------
```

Execution structu

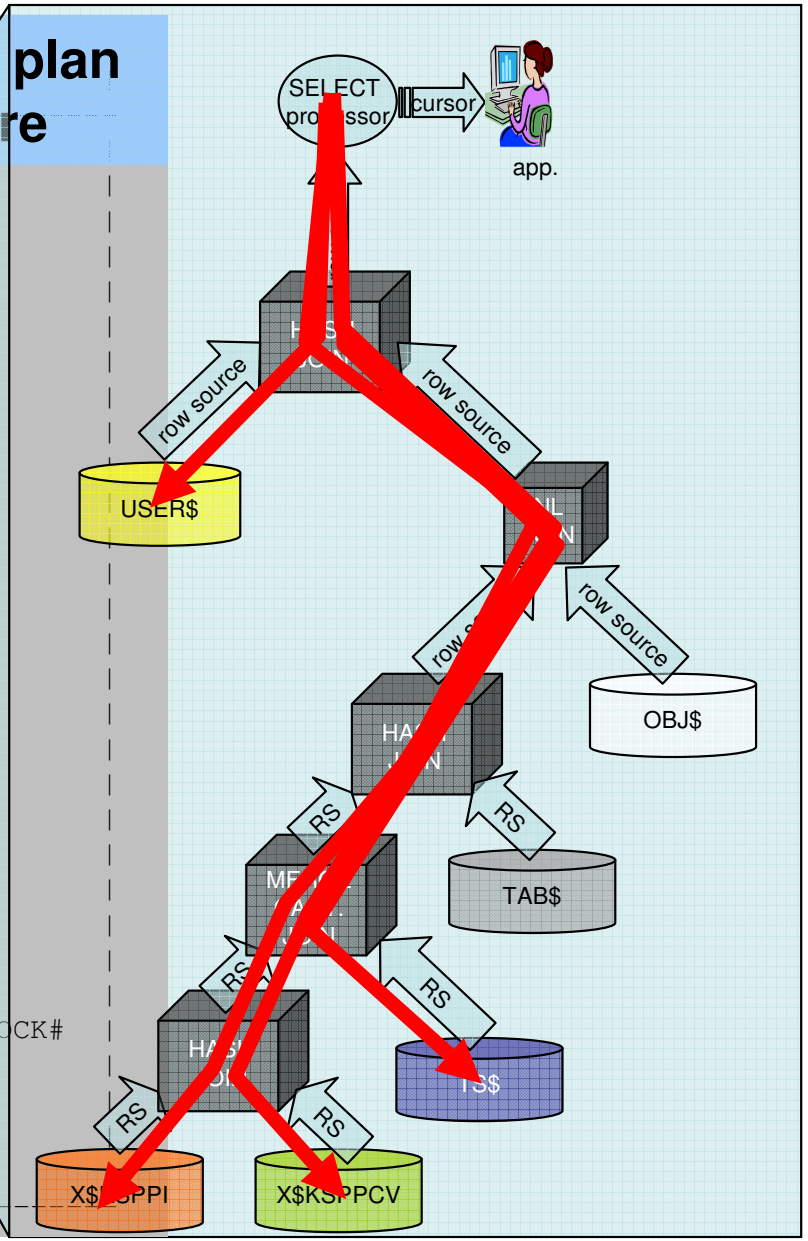


Row source execution

- *At any point of time only one branch of exec. plan is executed*

```
-----------------------------------------------------------------------
| Id  | Operation                                | Name                 |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                      |
|*  1 |   FILTER                                 |                      |
|   2 |     NESTED LOOPS OUTER                   |                      |
|*  3 |       HASH JOIN OUTER                    |                      |
|   4 |         NESTED LOOPS OUTER               |                      |
|   5 |           NESTED LOOPS OUTER             |                      |
|*  6 |             HASH JOIN                    |                      |
|   7 |               TABLE ACCESS FULL          | USER$                |
|   8 |               NESTED LOOPS               |                      |
|*  9 |                 HASH JOIN                |                      |
|  10 |                   MERGE JOIN CARTESIAN   |                      |
|* 11 |                     HASH JOIN            |                      |
|* 12 |                       FIXED TABLE FULL   | X$KSPPI              |
|  13 |                       FIXED TABLE FULL   | X$KSPPCV             |
|  14 |                     BUFFER SORT          |                      |
|  15 |                       TABLE ACCESS FULL  | TS$                  |
|* 16 |                   TABLE ACCESS FULL      | TAB$                 |
|* 17 |                 TABLE ACCESS BY INDEX ROWID | OBJ$                 |
|* 18 |                   INDEX UNIQUE SCAN      | I_OBJ1               |
|  19 |               TABLE ACCESS BY INDEX ROWID | OBJ$                 |
|* 20 |                 INDEX UNIQUE SCAN        | I_OBJ1               |
|  21 |             TABLE ACCESS BY INDEX ROWID  | OBJ$                 |
|* 22 |               INDEX UNIQUE SCAN          | I_OBJ1               |
|  23 |           TABLE ACCESS FULL              | USER$                |
|  24 |         TABLE ACCESS CLUSTER             | SEG$                 |
|* 25 |           INDEX UNIQUE SCAN              | I_FILE#_BLOCK#       |
|  26 |       NESTED LOOPS                       |                      |
|* 27 |         INDEX RANGE SCAN                 | I_OBJAUTH1           |
|* 28 |         FIXED TABLE FULL                 | X$KZSRO              |
|* 29 |           FIXED TABLE FULL               | X$KZSPR              |
-----------------------------------------------------------------------
```

Execution plan structure



Reading SQL plan execution stack

```
$ pstack 1354 | os_explain
   SELECT FETCH:
    SORT: Fetch
     HASH JOIN: Fetch
    * HASH JOIN: Fetch
     * VIEW: Fetch
        NESTED LOOP OUTER: Fetch
         NESTED LOOP OUTER: Fetch
          NESTED LOOP JOIN: Fetch
           HASH JOIN: Fetch
          * VIEW: Fetch
             UNION-ALL: Fetch
            * VIEW: Fetch
               UNION-ALL: Fetch
              * NESTED LOOP OUTER: Fetch
                 NESTED LOOP OUTER: Fetch
                  GRANULE ITERATOR: Fetch
                   INDEX: FetchFastFullScan
                    kdirfrs
```

Interpreting rowsource functions with os_explain

```
select /*+ ordered use_nl(b) use_nl(c) use_nl(d)
           full(a) full(b) full(c) full(d) */
count(*)
from    sys.obj$ a, sys.obj$ b, sys.obj$ c, sys.obj$ d
where
    a.owner# = b.owner# and b.owner# = c.owner#
and c.owner# = d.owner# and rownum <= 10000000000
```

```
-------------------------------------------
| Id  | Operation                  | Name |
-------------------------------------------
|   1 |  SORT AGGREGATE            |      |
|*  2 |   COUNT STOPKEY            |      |
|   3 |    NESTED LOOPS            |      |
|   4 |     NESTED LOOPS           |      |
|   5 |      NESTED LOOPS          |      |
|   6 |       TABLE ACCESS FULL    | OBJ$ |
|*  7 |       TABLE ACCESS FULL    | OBJ$ |
|*  8 |      TABLE ACCESS FULL     | OBJ$ |
|*  9 |     TABLE ACCESS FULL      | OBJ$ |
-------------------------------------------
```

```
$ pstack 1595 | ./os_explain
  kpoal8
   SELECT FETCH:
    GROUP BY SORT: Fetch
     COUNT: Fetch
      NESTED LOOP JOIN: Fetch
       TABLE ACCESS: Fetch
        kdsttgr
         kdstf0100101km
          expeal
           expepr
```

Child row source function must map directly to a child line in exec plan

Diagnosing PL/SQL execution

- V$SESSION new columns from 10.2.0.3
  - PLSQL_ENTRY_OBJECT_ID
  - PLSQL_ENTRY_SUBPROGRAM_ID
  - PLSQL_OBJECT_ID
  - PLSQL_SUBPROGRAM_ID

- Dump errorstack shows PL/SQL line number info:

```
*** 2008-12-05 11:43:17.781
ksedmp: internal or fatal error
Current SQL statement for this session:
BEGIN dbms_lock.sleep(100); END;
----- PL/SQL Call Stack -----
  object      line  object
  handle    number  name
20496E0C       201  package body SYS.DBMS_LOCK
204E46E8         1  anonymous block
```

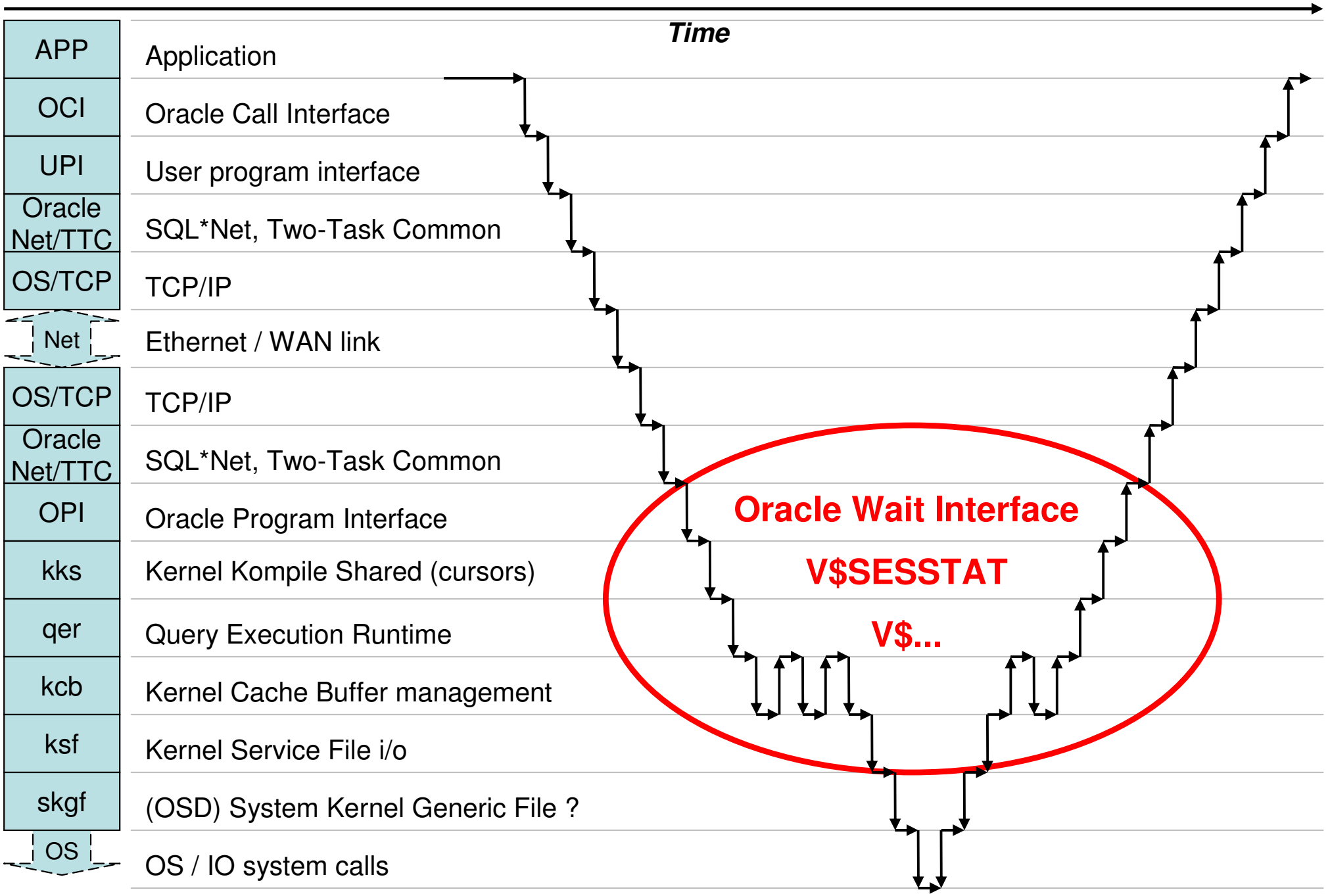
Scripts for low-level Oracle diagnosis - demos

- dstackprof
  - DTrace based stack profiler
- ostackprof
  - oradebug short_stack based stack profiler
- latchprofx
  - session level latch holder profiling script
  - plain SQL!
  - includes info in which function in Oracle kernel the latch get was invoked

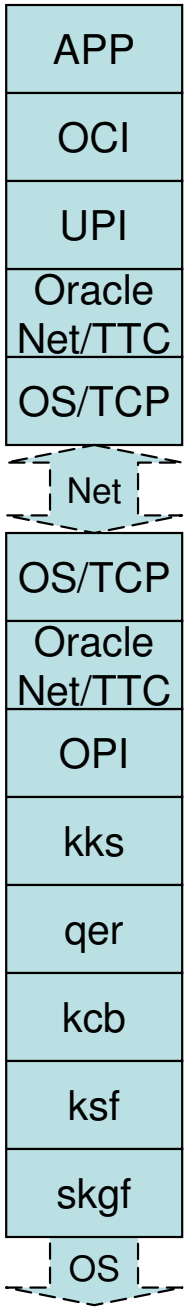
What if my problem lies outside Oracle?

…Where to look next?

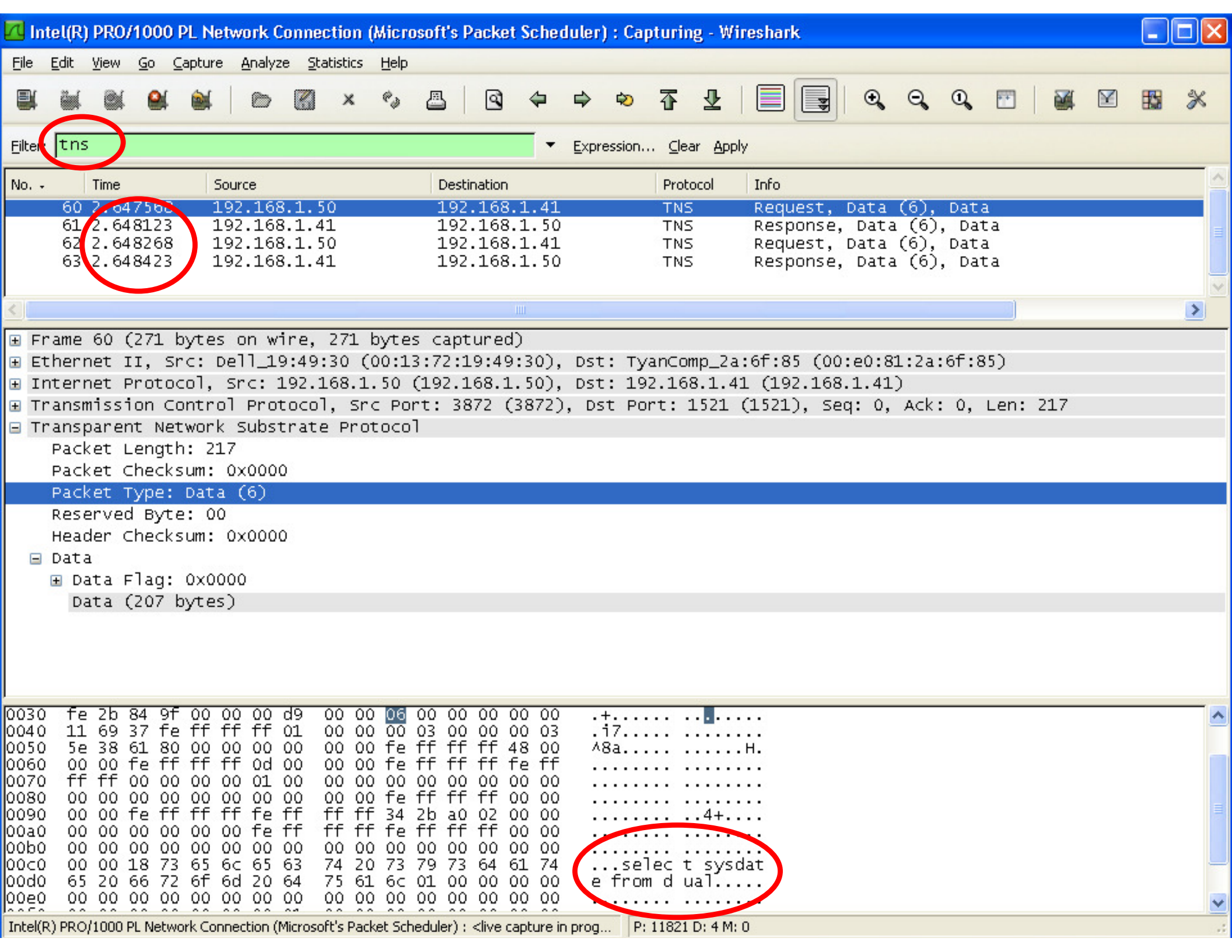
Oracle internal process flow



Oracle internal process flow



Application	***Application instrumentation***, ltrace, truss -u"libclntsh:*"
Oracle Call Interface	$OH/rdbms/demo/ociucb.mk, OCITrace
User program interface	-
SQL*NET, TNS, Two-Task Common	SQL*Net trace, Wireshark TNS protocol digester
TCP/IP	Wireshark TCP protocol digester, truss, strace
Ethernet / WAN link	snoop, tcpdump, Wireshark
TCP/IP	Wireshark TCP protocol digester, truss, strace
SQL*NET, TNS, Two-Task Common	SQL*Net trace, Wireshark, Event 10079
Oracle Program Interface	Event 10051
Kernel Kompile Shared (cursors)	sql_trace, Event 10046, 10270, cursortrace
Query Execution Runtime	v$sql_plan_statistics, v$sql_plan_statistics_all, sql_trace
Kernel Cache Buffer management	x$kcbsw, Event 10200,10812, _trace_pin_time
Kernel Service File i/o	v$filestat, v$tempstat, v$session_wait, Event 10298
(OSD) System Kernel Generic File ?	-
OS / IO system calls	DTrace, strace, truss, tusc, filemon.exe, procmon.exe


Intel(R) PRO/1000 PL Network Connection (Microsoft's Packet Scheduler) : Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: tns
Expression... Clear Apply
No. Time Source Destination Protocol Info
60 2.647562 192.168.1.50 192.168.1.41 TNS Request, Data (6), Data
61 2.648123 192.168.1.41 192.168.1.50 TNS Response, Data (6), Data
62 2.648268 192.168.1.50 192.168.1.41 TNS Request, Data (6), Data
63 2.648423 192.168.1.41 192.168.1.50 TNS Response, Data (6), Data
Frame 60 (271 bytes on wire, 271 bytes captured)
Ethernet II, Src: Dell_19:49:30 (00:13:72:19:49:30), Dst: TyanComp_2a:6f:85 (00:e0:81:2a:6f:85)
Internet Protocol, Src: 192.168.1.50 (192.168.1.50), Dst: 192.168.1.41 (192.168.1.41)
Transmission Control Protocol, Src Port: 3872 (3872), Dst Port: 1521 (1521), Seq: 0, Ack: 0, Len: 217
Transparent Network Substrate Protocol
Packet Length: 217
Packet Checksum: 0x0000
Packet Type: Data (6)
Reserved Byte: 00
Header Checksum: 0x0000
Data
Data Flag: 0x0000
Data (207 bytes)
0030 fe 2b 84 9f 00 00 00 d9 00 00 06 00 00 00 00 00 .+...... ..
0040 11 69 37 fe ff ff ff 01 00 00 00 03 00 00 00 03 .i7..... ....
0050 5e 38 61 80 00 00 00 00 00 00 fe ff ff ff 48 00 ^8a..... ....H.
0060 00 00 fe ff ff ff 0d 00 00 00 fe ff ff ff fe ff ....
0070 ff ff 00 00 00 00 01 00 00 00 00 00 00 00 00 00 ....
0080 00 00 00 00 00 00 00 00 00 00 fe ff ff ff 00 00 ....
0090 00 00 fe ff ff ff fe ff ff ff 34 2b a0 02 00 00 ....4+....
00a0 00 00 00 00 00 00 fe ff ff ff fe ff ff ff 00 00 ....
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00c0 00 00 18 73 65 6c 65 63 74 20 73 79 73 64 61 74 ...selec t sysdat
00d0 65 20 66 72 6f 6d 20 64 75 61 6c 01 00 00 00 00 e from d ual.....
00e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Intel(R) PRO/1000 PL Network Connection (Microsoft's Packet Scheduler) : <live capture in prog...
P: 11821 D: 4 M: 0

Repeat: Session troubleshooting sequence

1. Oracle Wait Interface - *response **TIME!***

?

2. v$sesstat performance counters - *hints, indicators*

?

3. Process stack - *truth about process execution*

!!!

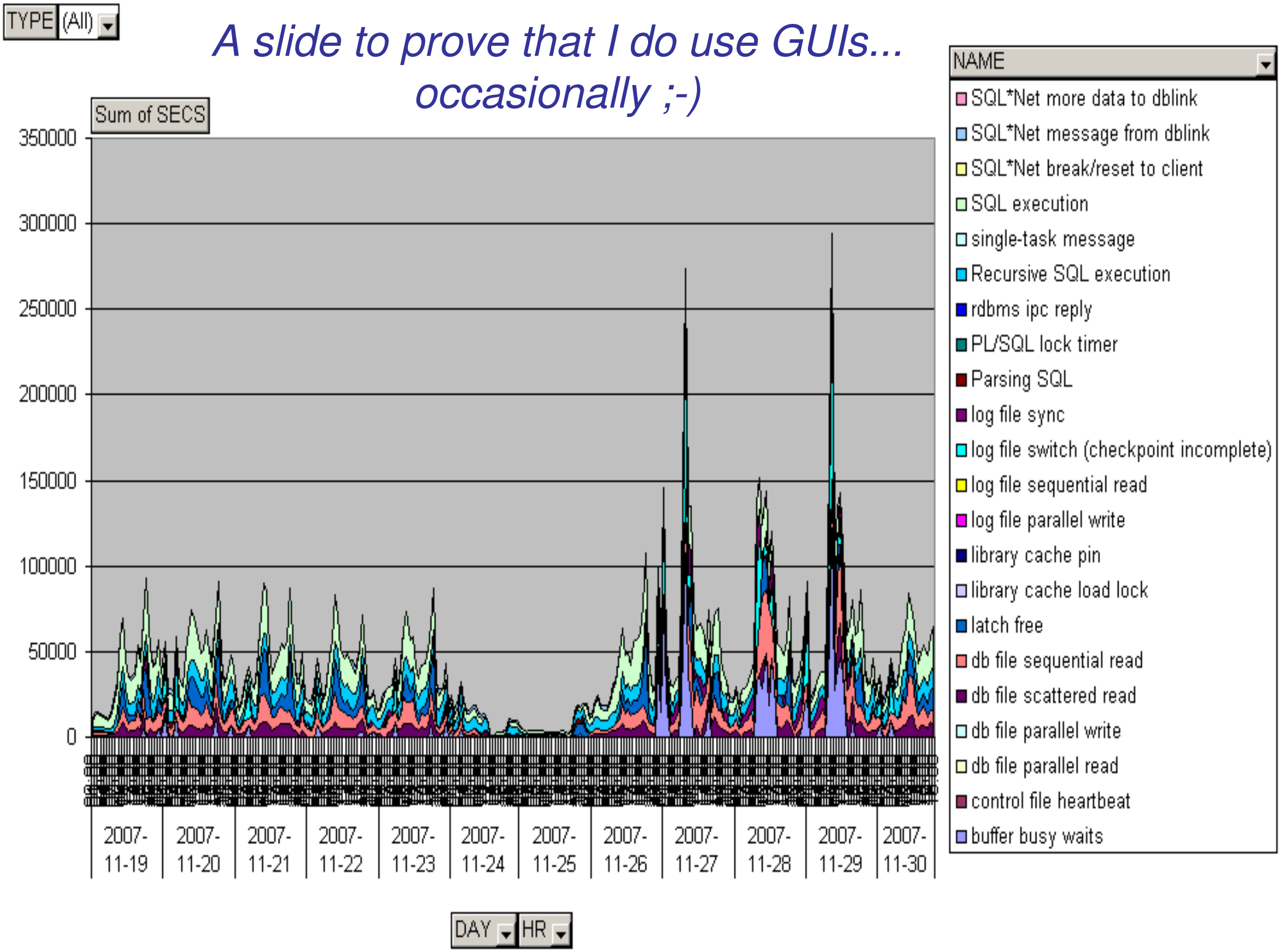
Session-level troubleshooting tools

1. Profile session wait / CPU breakdown - ***TIME!***
   - *V$SESSION_WAIT*
   - *V$SESSION_EVENT, V$SESS_TIME_MODEL (10g+)*
   - *Snapper, Sesspack, Statspack session mode*

2. Profile session performance counters - *occurrences*
   - *V$SESSTAT*

3. Profile session's process stack - *statistical sample*
   - *pstack, procstack, ostackprof, dstackprof*
   - *procmon.exe, procexp.exe*
   - *oradebug short_stack, oradebug errorstack*

A slide to prove that I do use GUIs... occasionally ;-)



References

- Metalink notes:
  - 459694.1 - Procwatcher: Script to Monitor and Examine Oracle and CRS Processes
  - 175982.1 - ORA-600 Lookup Error Categories (explains function names)
- Web:
  - http://blog.tanelpoder.com
  - http://www.juliandyke.com/Diagnostics/Diagnostics.html
- Seminar:
  - If you like this stuff, you'll definitely like my seminar!
  - http://blog.tanelpoder.com/seminar/

Questions?

Further questions welcome at
http://blog.tanelpoder.com

Thank you!

Tanel Põder

tanel@tanelpoder.com

http://www.tanelpoder.com