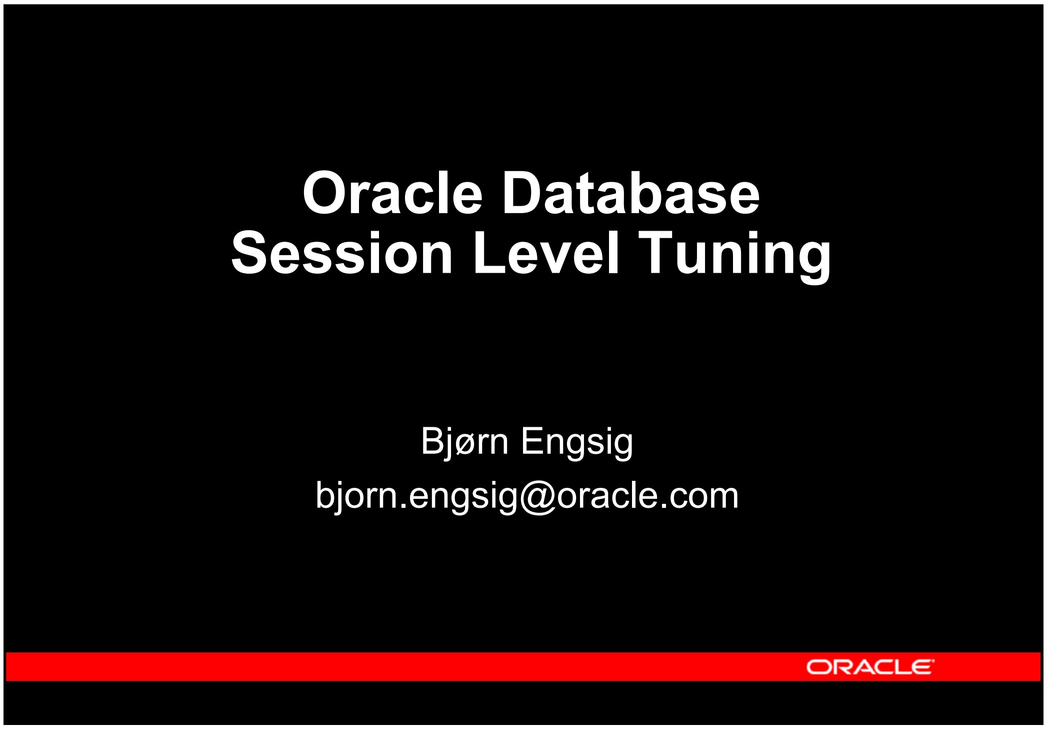
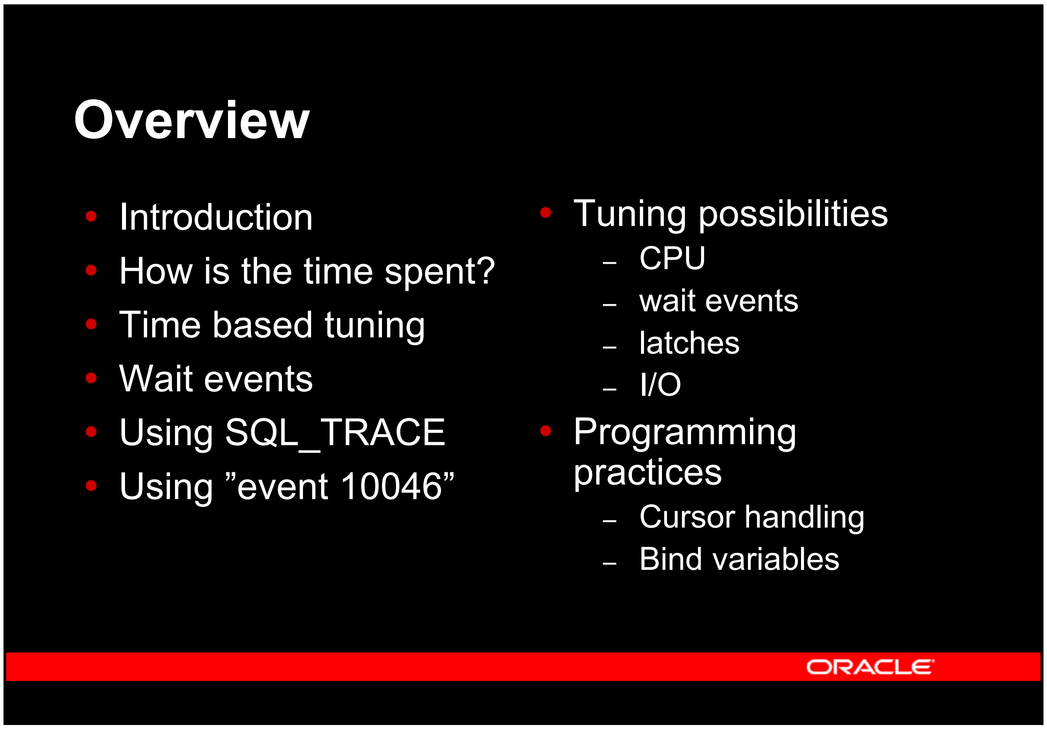


Oracle Database Session Level Tuning

Bjørn Engsig
bjorn.engsig@oracle.com



Overview

- Introduction
- How is the time spent?
- Time based tuning
- Wait events
- Using SQL_TRACE
- Using ”event 10046”
- Tuning possibilities
  - CPU
  - wait events
  - latches
  - I/O
- Programming practices
  - Cursor handling
  - Bind variables

Some typical performance questions

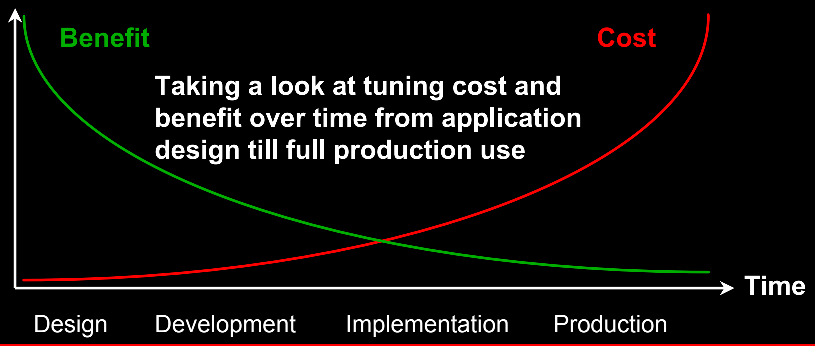
- Why is database performance ALWAYS a hot topic?
- Why does my application not scale?
- Where does my performance problem really come from?
- Can I set a magic init.ora parameter?
  - There are in fact some, although not magical ones!

This presentation focuses on *SQL* processing - rather than on *data* processing

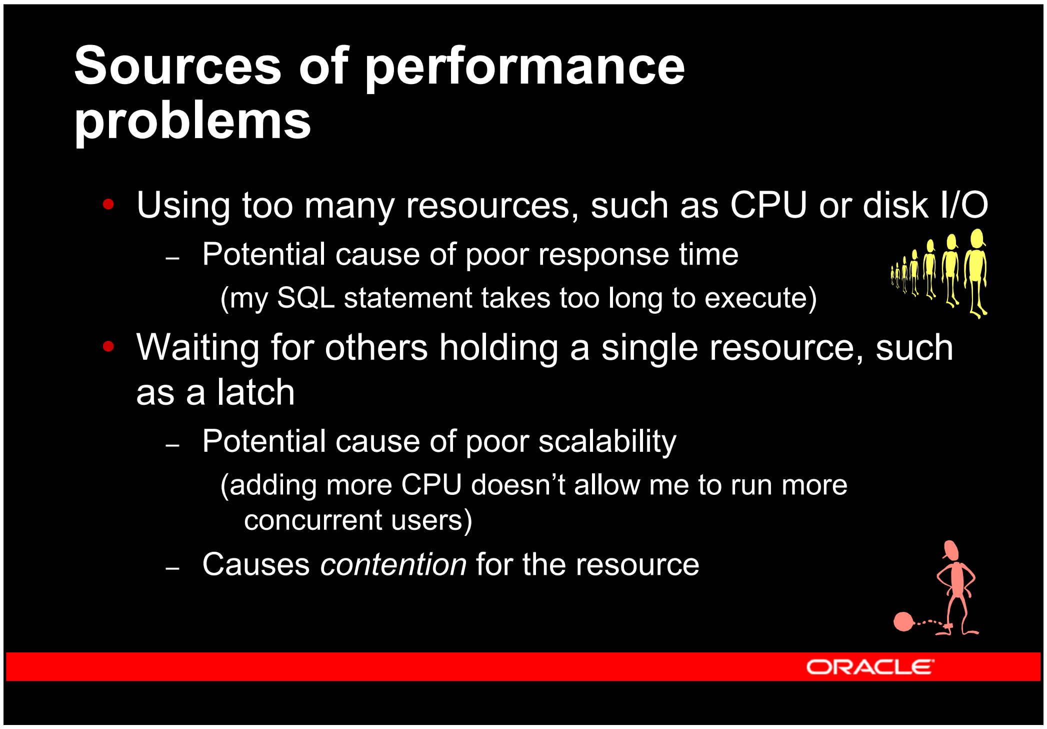
A famous picture

Tuning cost increases in time

Tuning benefit decreases in time



Taking a look at tuning cost and benefit over time from application design till full production use

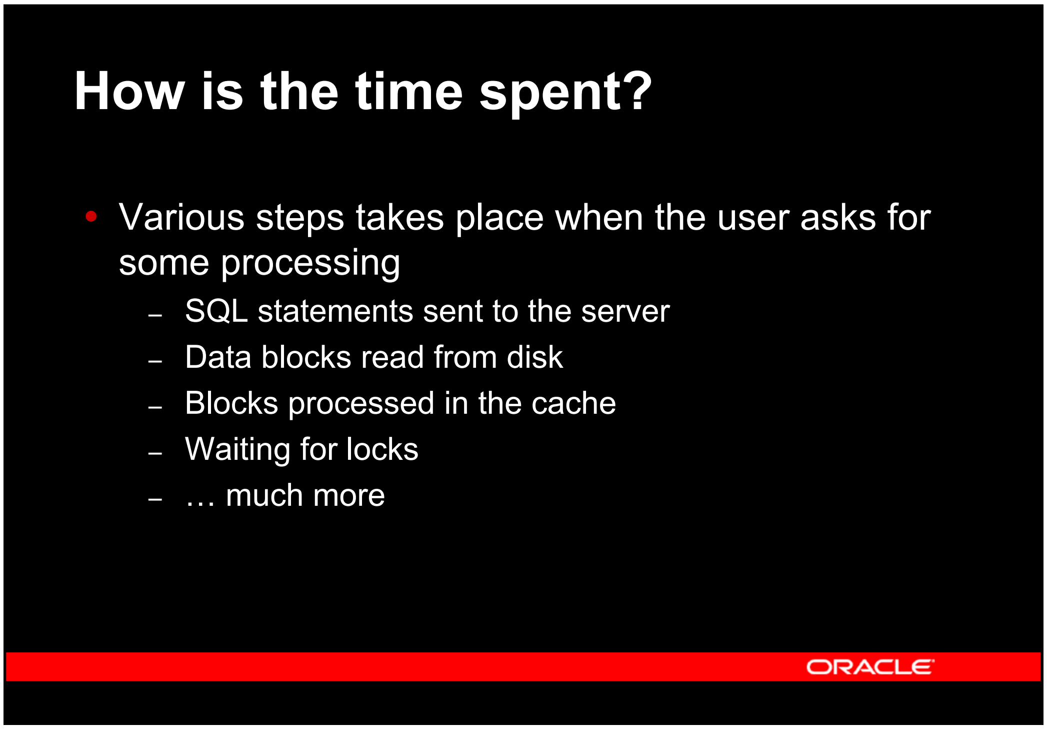


Sources of performance problems

- Using too many resources, such as CPU or disk I/O
  - Potential cause of poor response time

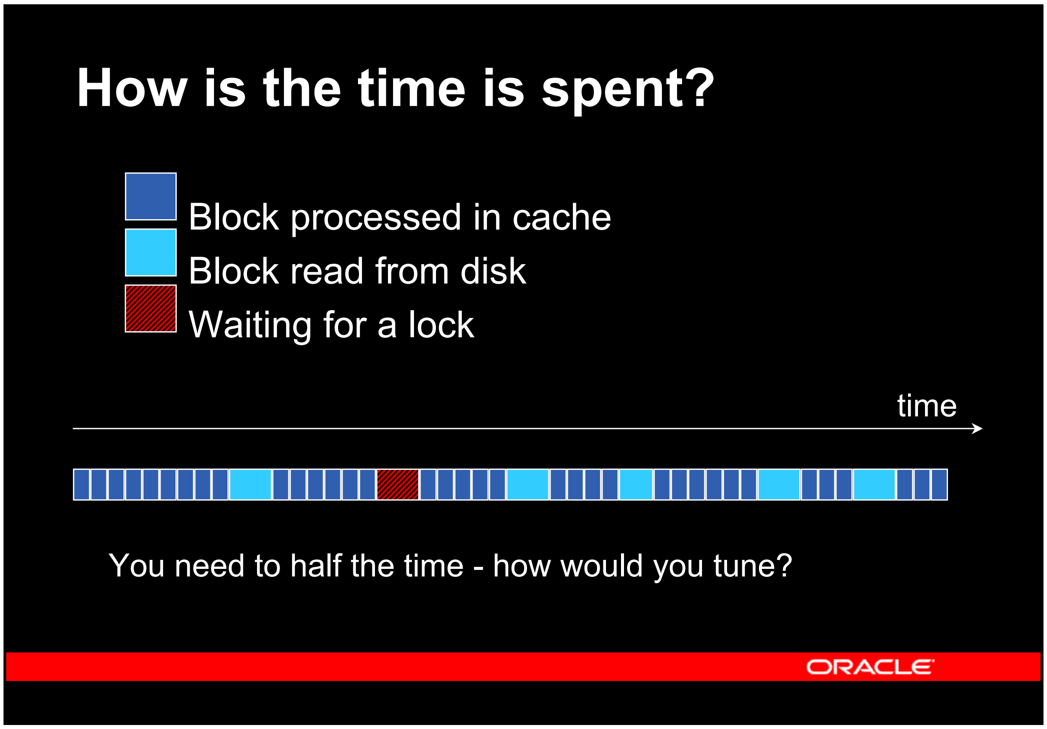
    (my SQL statement takes too long to execute)
- Waiting for others holding a single resource, such as a latch
  - Potential cause of poor scalability

    (adding more CPU doesn’t allow me to run more concurrent users)
  - Causes *contention* for the resource



How is the time spent?

- Various steps takes place when the user asks for some processing
  - SQL statements sent to the server
  - Data blocks read from disk
  - Blocks processed in the cache
  - Waiting for locks
  - … much more



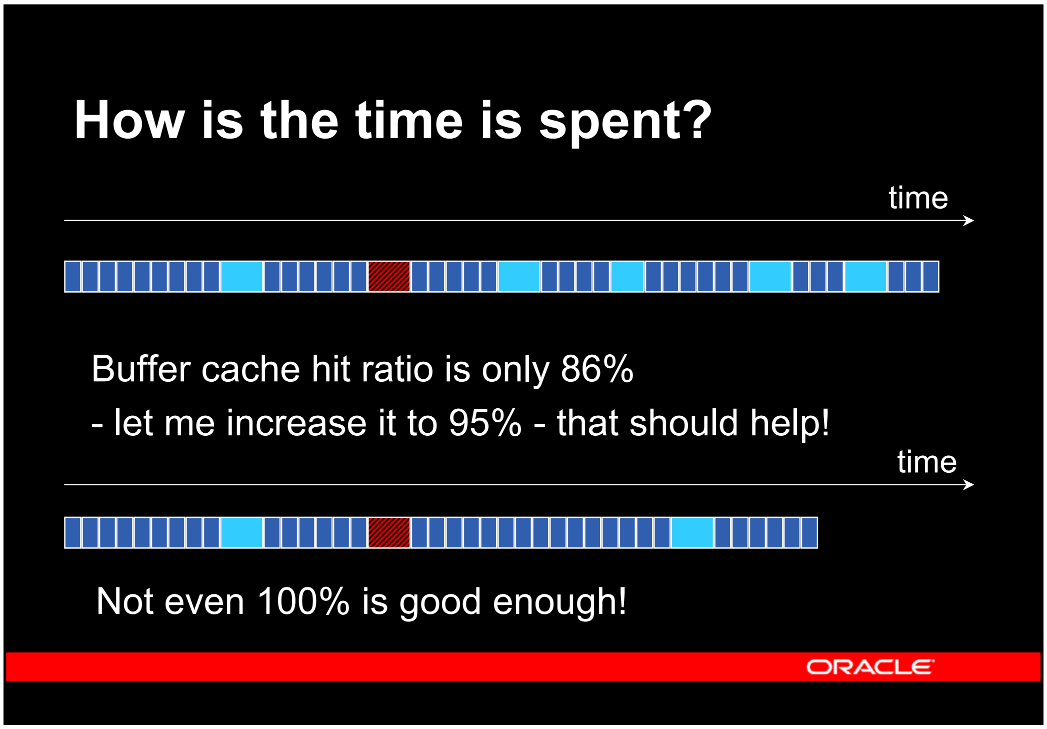
How is the time is spent?

■ Block processed in cache

□ Block read from disk

▨ Waiting for a lock

You need to half the time - how would you tune?

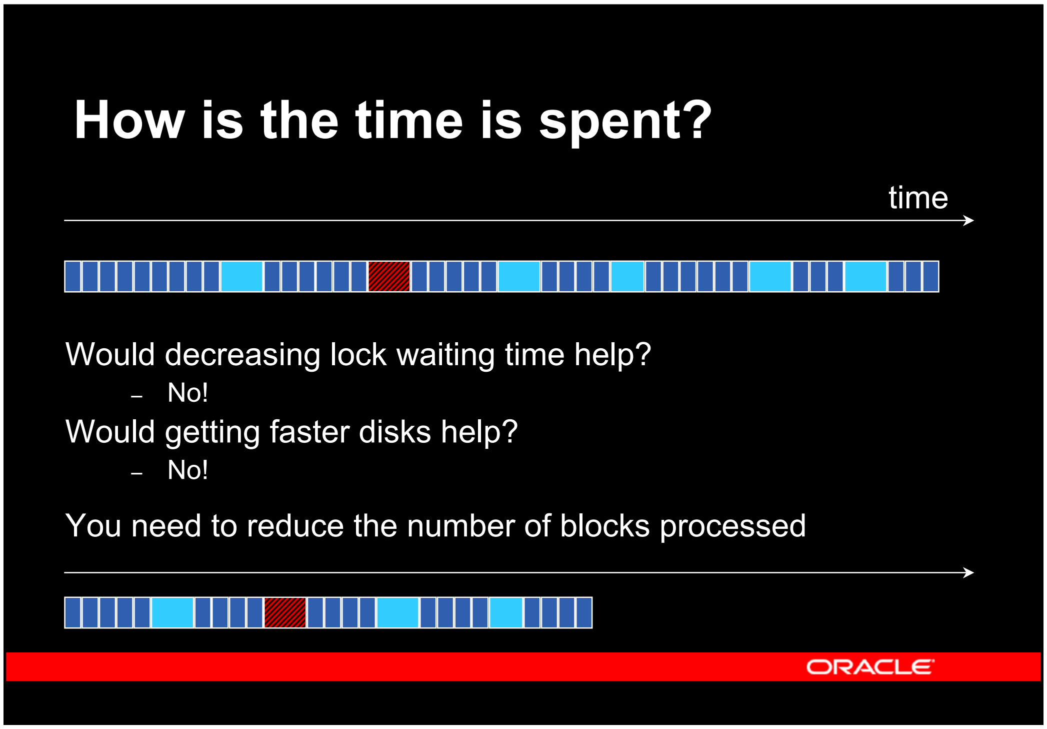


How is the time is spent?

Buffer cache hit ratio is only 86%

- let me increase it to 95% - that should help!

Not even 100% is good enough!



How is the time is spent?

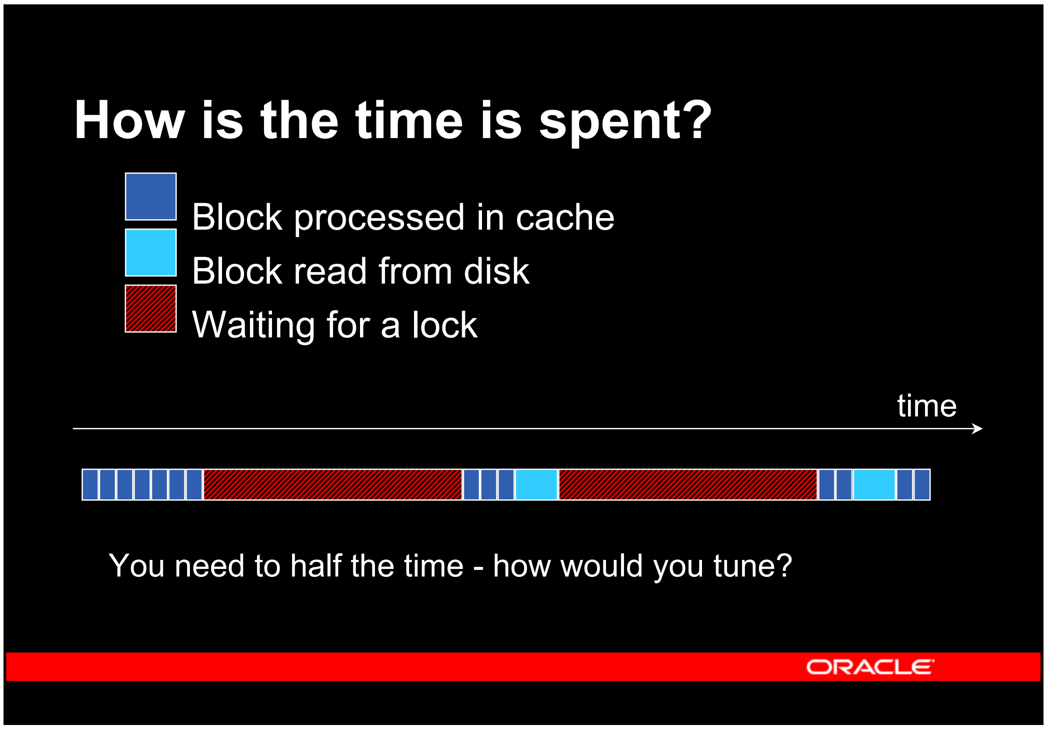
Would decreasing lock waiting time help?

- No!

Would getting faster disks help?

- No!

You need to reduce the number of blocks processed



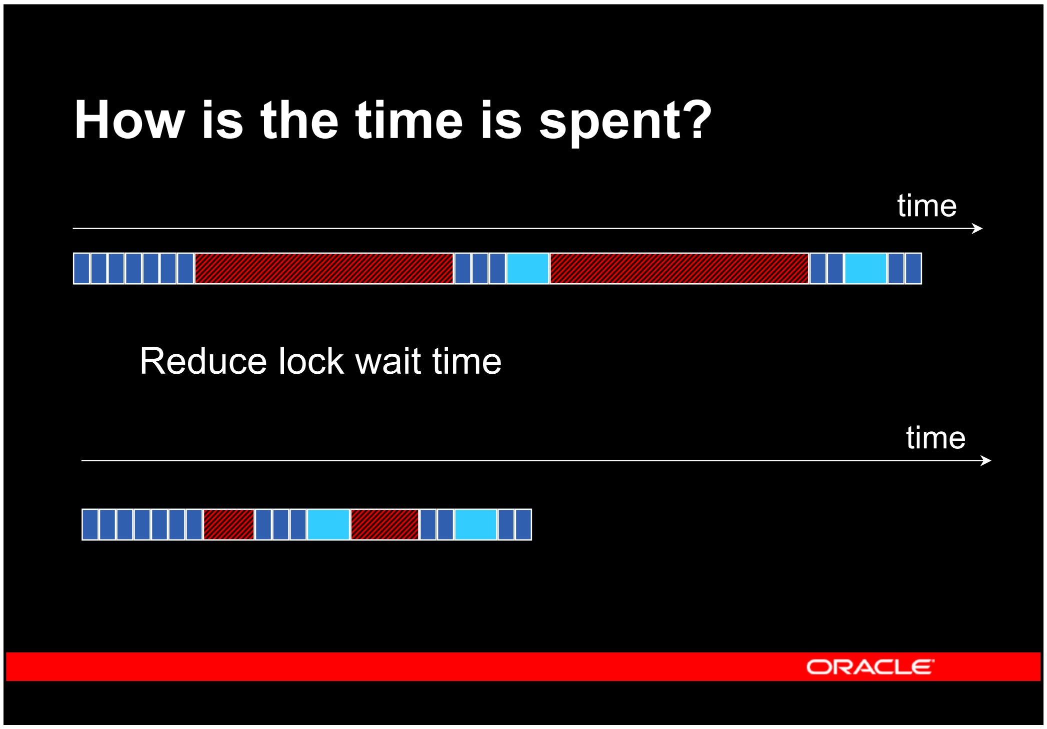
How is the time is spent?

■ Block processed in cache

□ Block read from disk

▨ Waiting for a lock

You need to half the time - how would you tune?



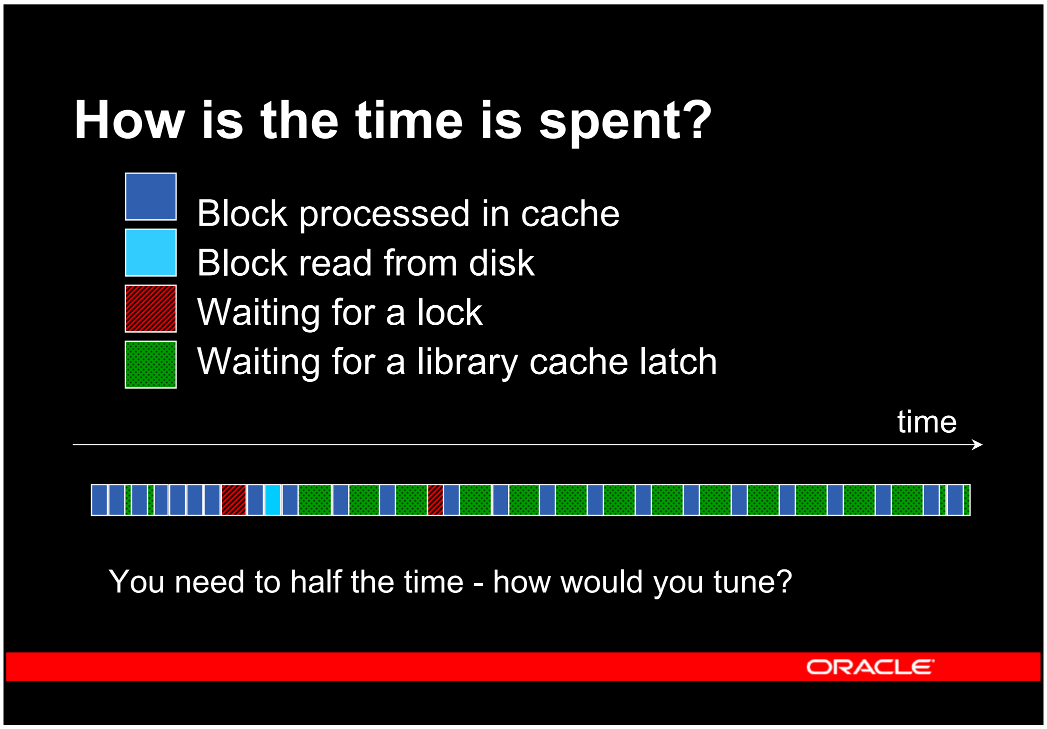
How is the time is spent?

time

Reduce lock wait time

time

ORACLE



How is the time is spent?

🟦 Block processed in cache

🟦 Block read from disk

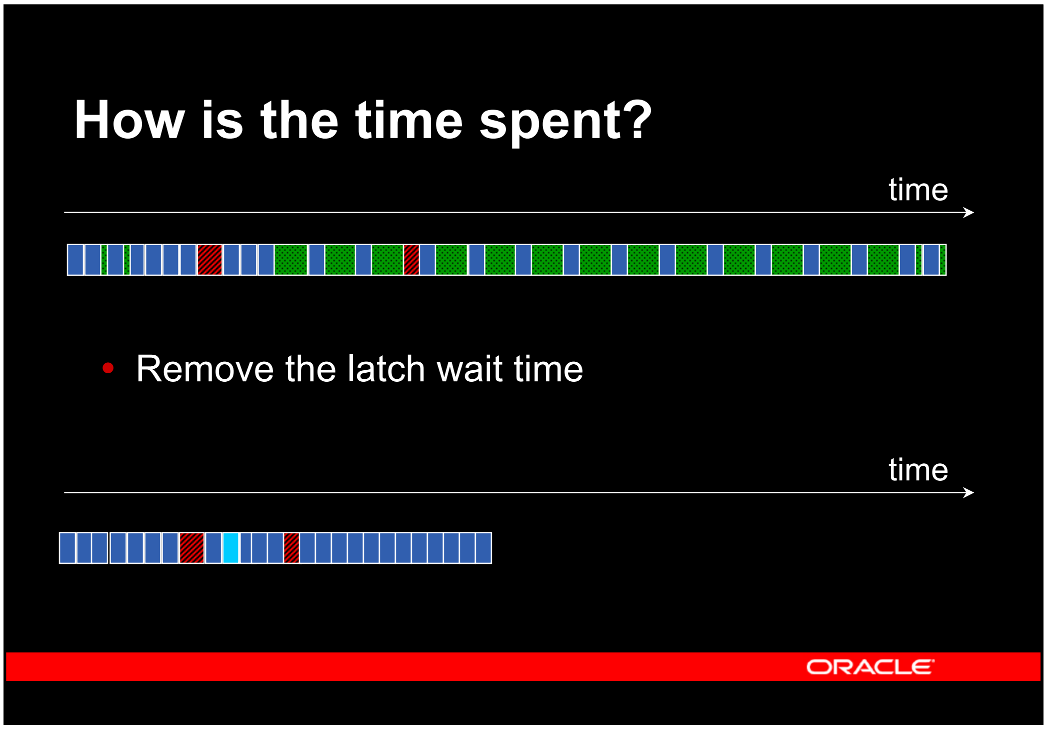
🟥 Waiting for a lock

🟩 Waiting for a library cache latch

time

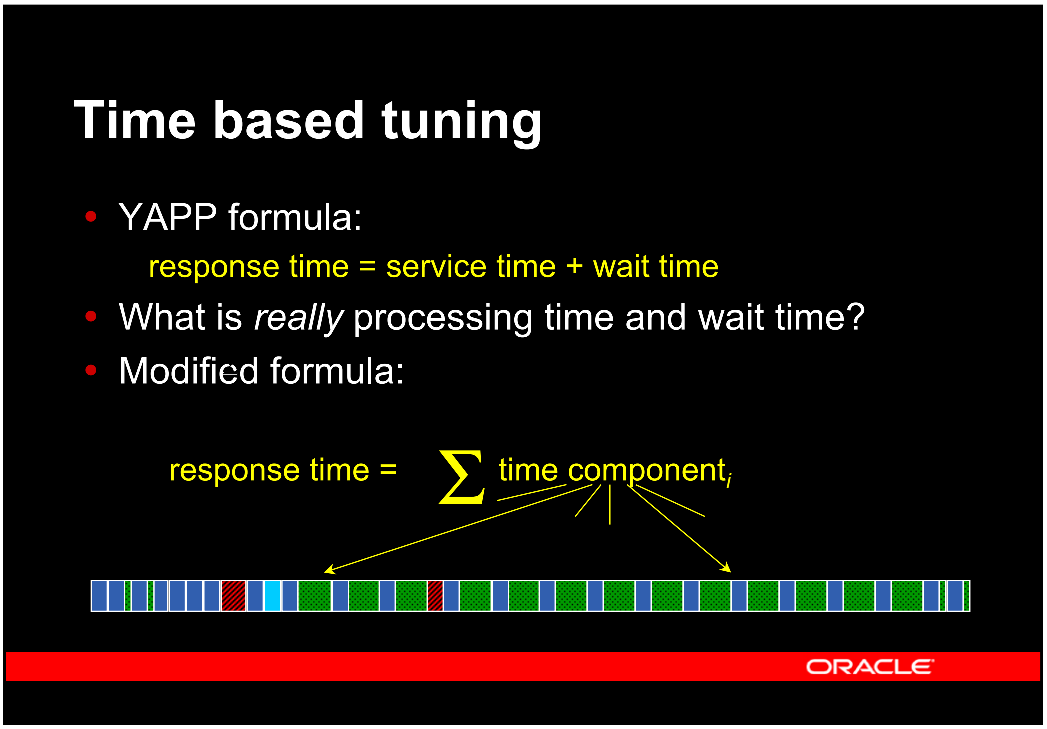
You need to half the time - how would you tune?

ORACLE



How is the time spent?

- Remove the latch wait time



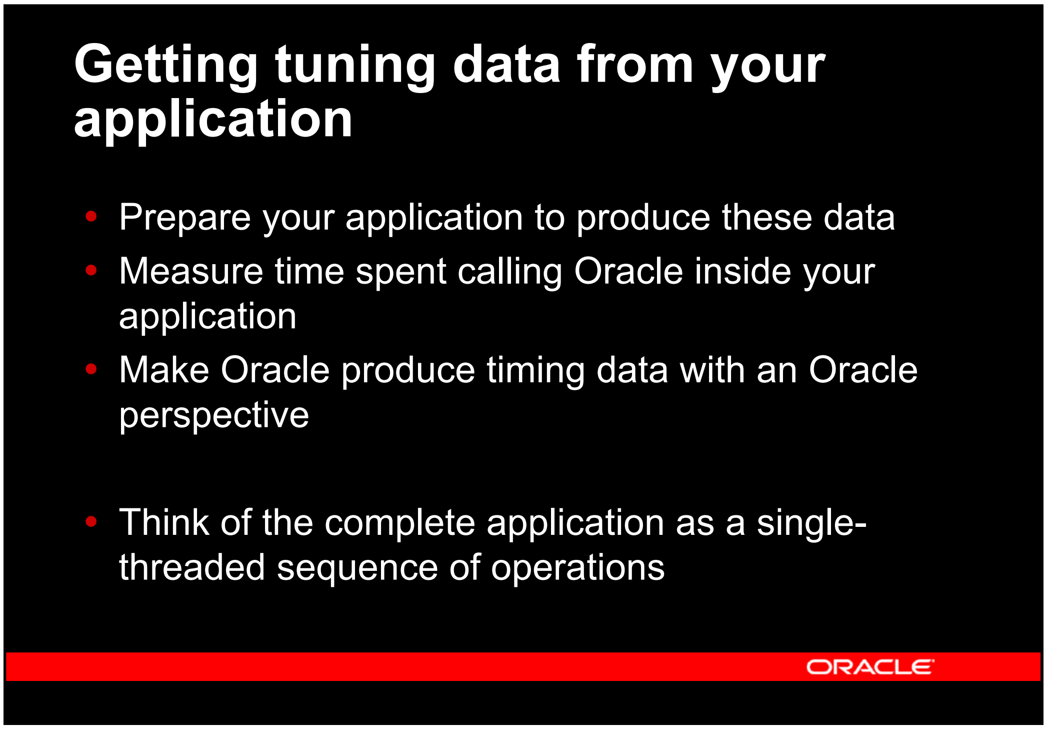
Time based tuning

- YAPP formula:

  response time = service time + wait time

- What is *really* processing time and wait time?
- Modified formula:

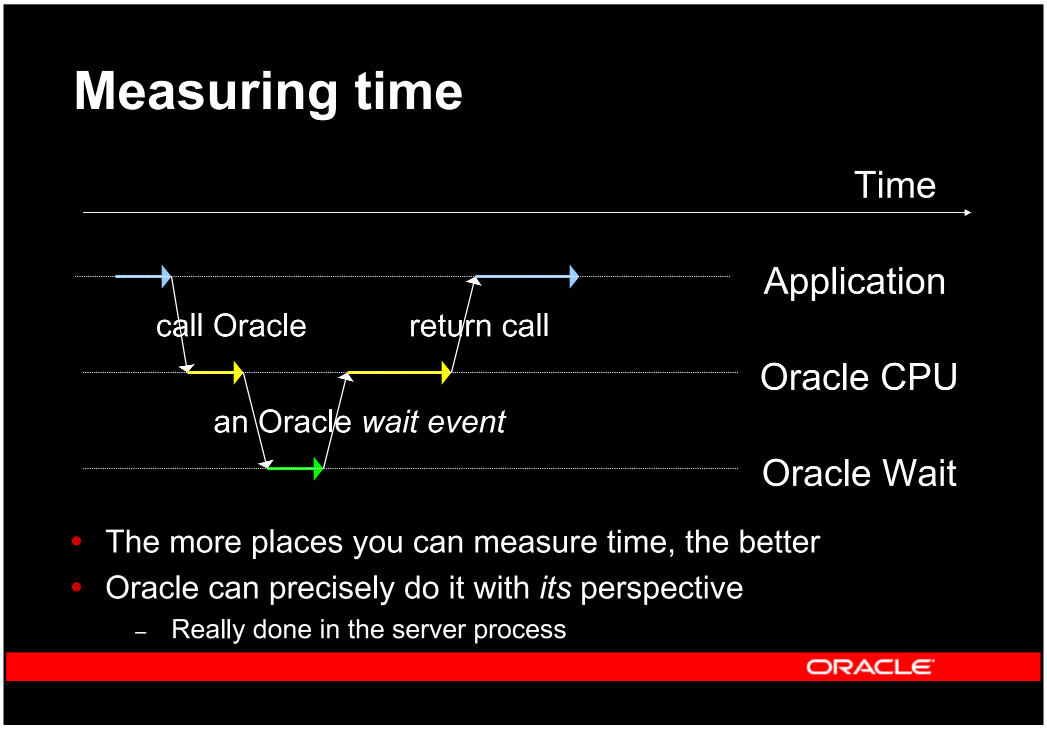
$$\text{response time} = \sum \text{time component}_i$$



Getting tuning data from your application

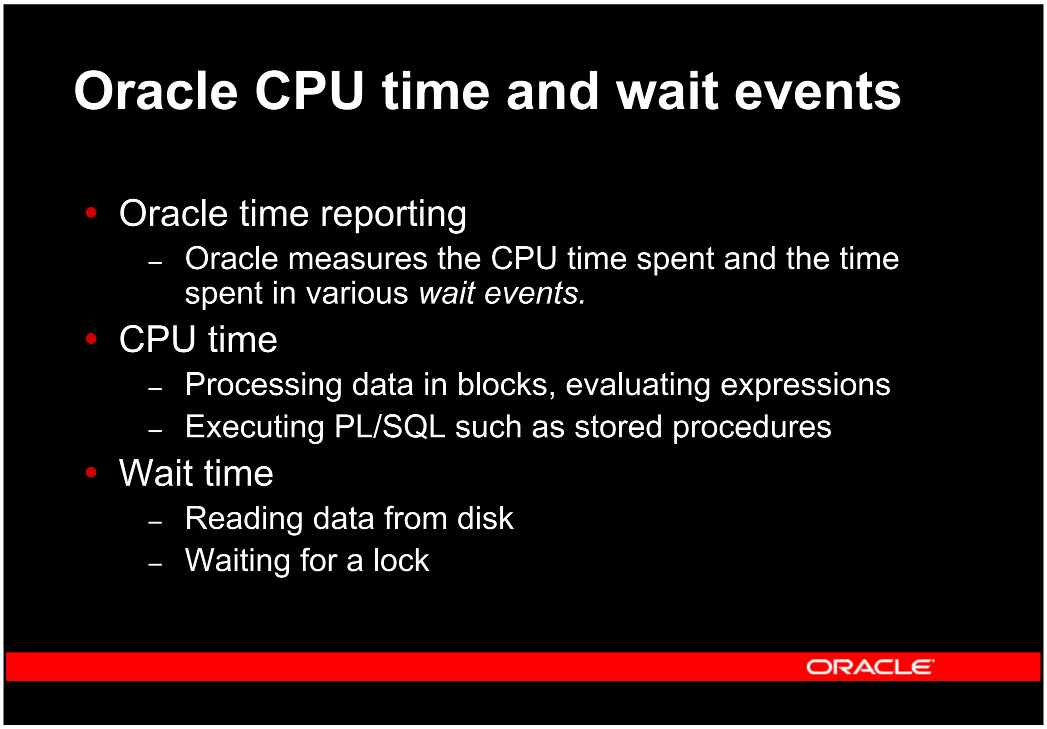
- Prepare your application to produce these data
- Measure time spent calling Oracle inside your application
- Make Oracle produce timing data with an Oracle perspective

- Think of the complete application as a single-threaded sequence of operations



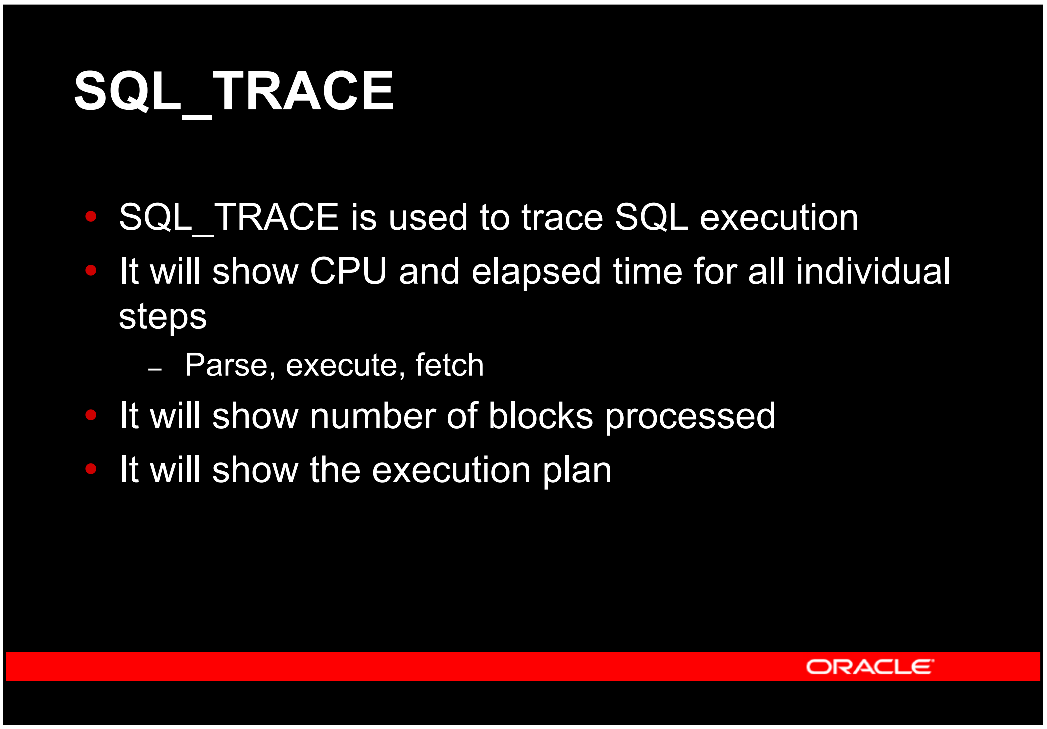
Measuring time

- The more places you can measure time, the better
- Oracle can precisely do it with *its* perspective
  - Really done in the server process



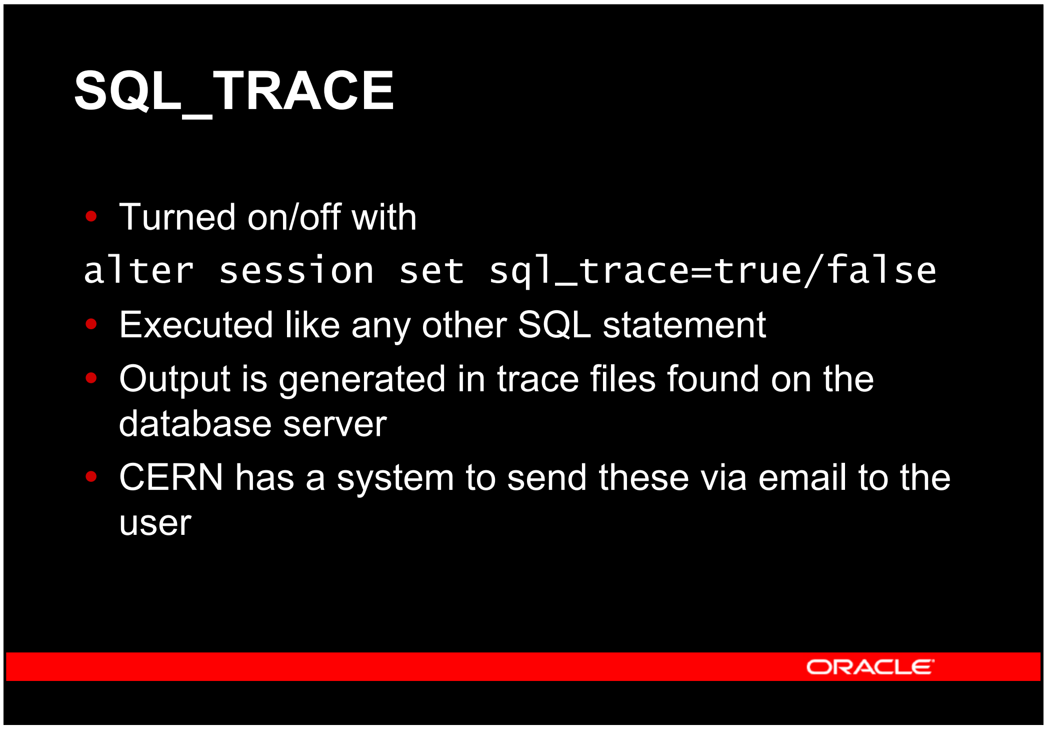
Oracle CPU time and wait events

- Oracle time reporting
  - Oracle measures the CPU time spent and the time spent in various *wait events.*
- CPU time
  - Processing data in blocks, evaluating expressions
  - Executing PL/SQL such as stored procedures
- Wait time
  - Reading data from disk
  - Waiting for a lock



SQL_TRACE

- SQL_TRACE is used to trace SQL execution
- It will show CPU and elapsed time for all individual steps
  - Parse, execute, fetch
- It will show number of blocks processed
- It will show the execution plan

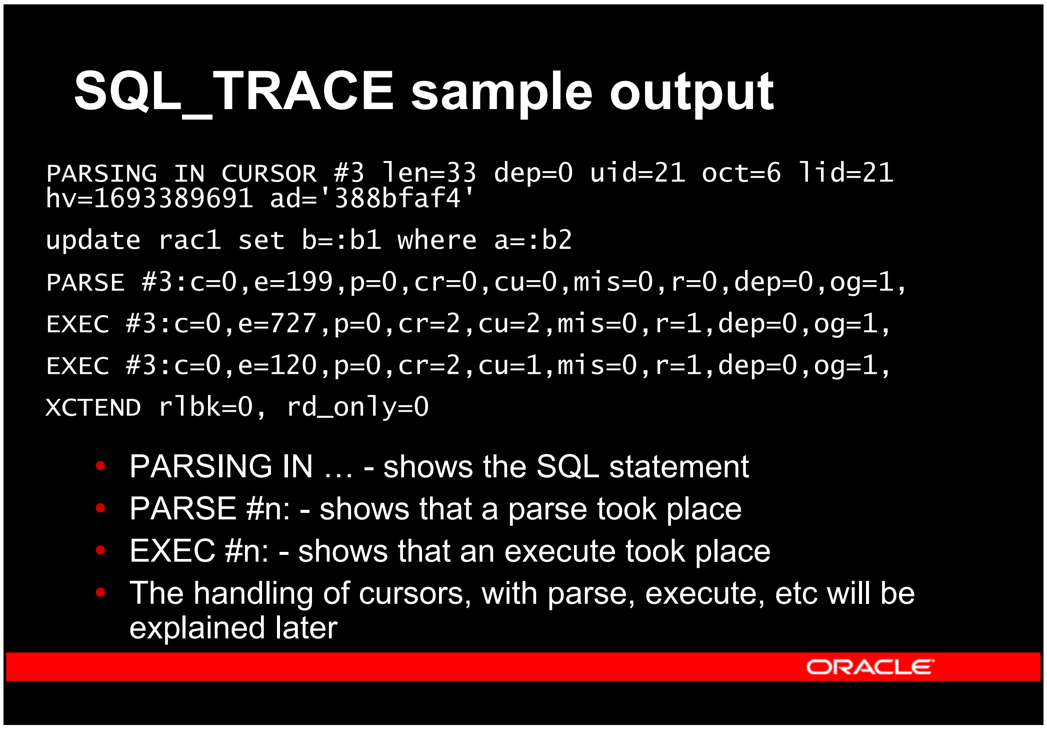


SQL_TRACE

- Turned on/off with

```
alter session set sql_trace=true/false
```

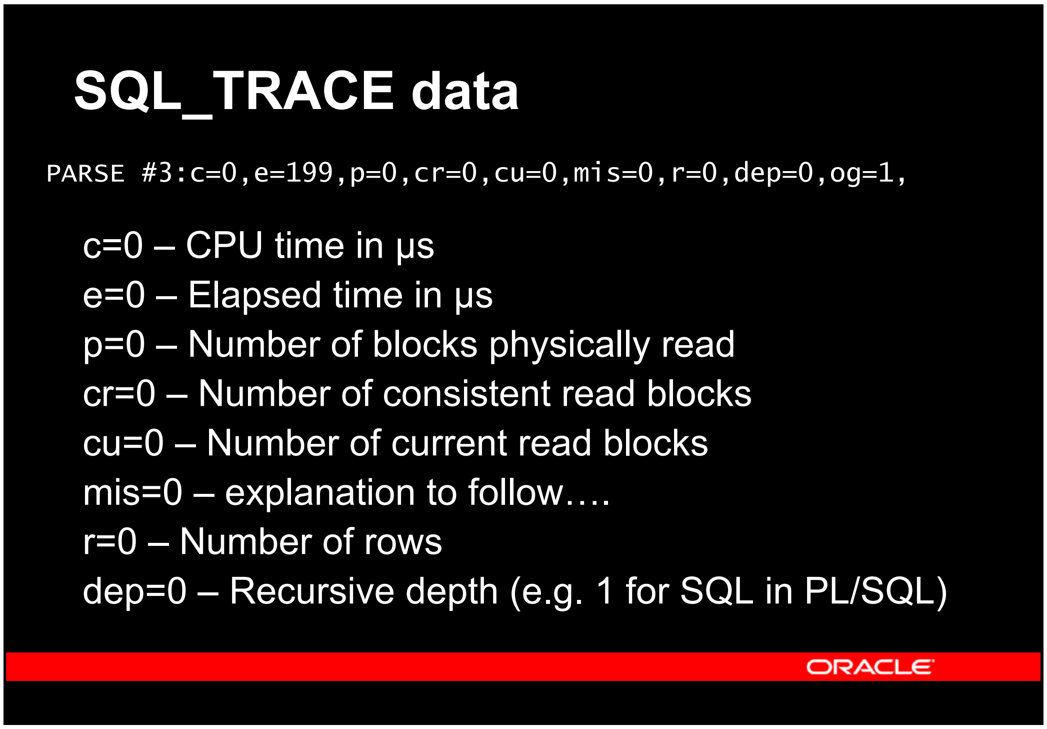
- Executed like any other SQL statement
- Output is generated in trace files found on the database server
- CERN has a system to send these via email to the user



SQL_TRACE sample output

```
PARSING IN CURSOR #3 len=33 dep=0 uid=21 oct=6 lid=21
hv=1693389691 ad='388bfaf4'
update rac1 set b=:b1 where a=:b2
PARSE #3:c=0,e=199,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,
EXEC #3:c=0,e=727,p=0,cr=2,cu=2,mis=0,r=1,dep=0,og=1,
EXEC #3:c=0,e=120,p=0,cr=2,cu=1,mis=0,r=1,dep=0,og=1,
XCTEND rlbk=0, rd_only=0
```

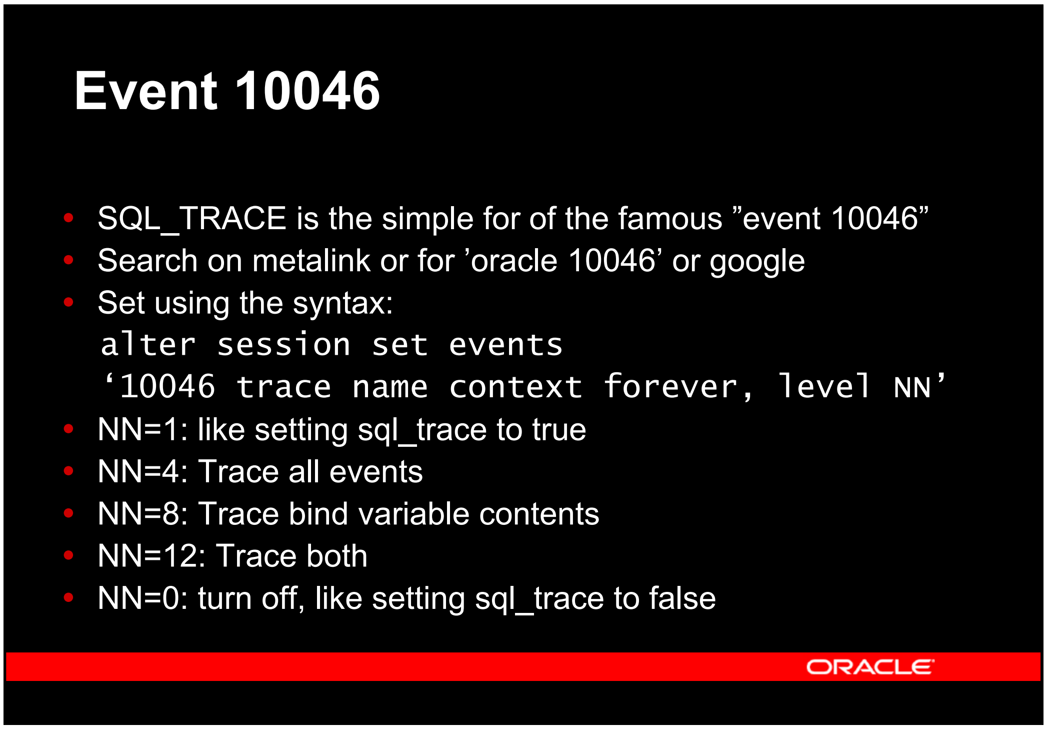
- PARSING IN … - shows the SQL statement
- PARSE #n: - shows that a parse took place
- EXEC #n: - shows that an execute took place
- The handling of cursors, with parse, execute, etc will be explained later



SQL_TRACE data

```
PARSE #3:c=0,e=199,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,
```

c=0 – CPU time in μs
e=0 – Elapsed time in μs
p=0 – Number of blocks physically read
cr=0 – Number of consistent read blocks
cu=0 – Number of current read blocks
mis=0 – explanation to follow….
r=0 – Number of rows
dep=0 – Recursive depth (e.g. 1 for SQL in PL/SQL)

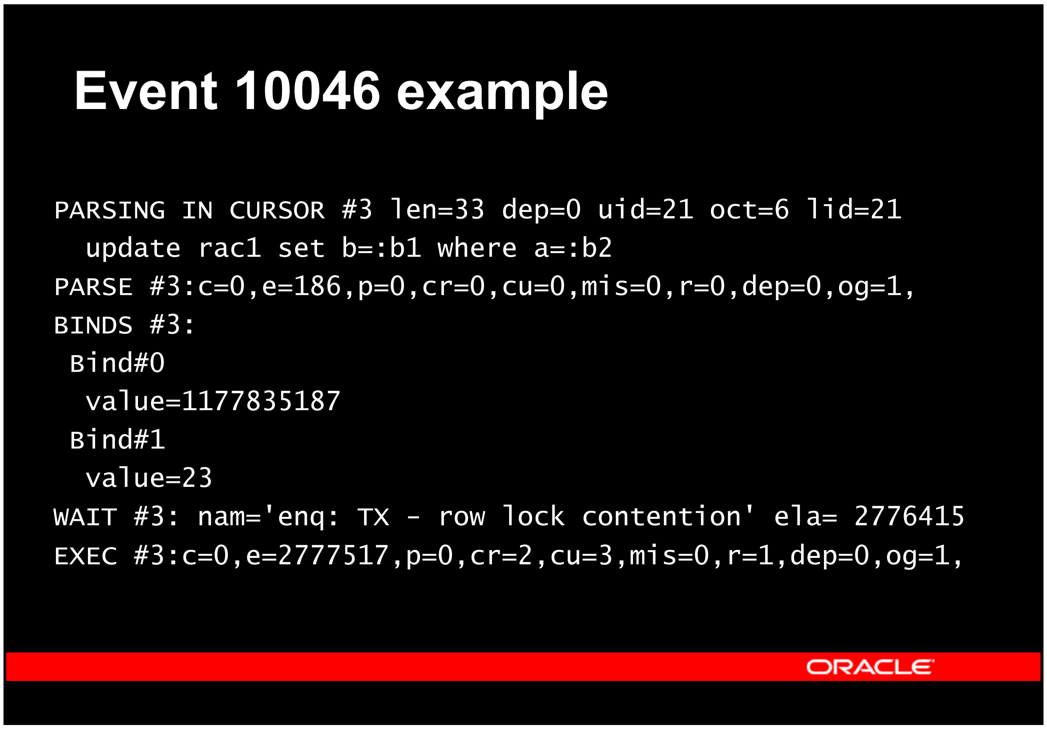


Event 10046

- SQL_TRACE is the simple for of the famous ”event 10046”
- Search on metalink or for ’oracle 10046’ or google
- Set using the syntax:

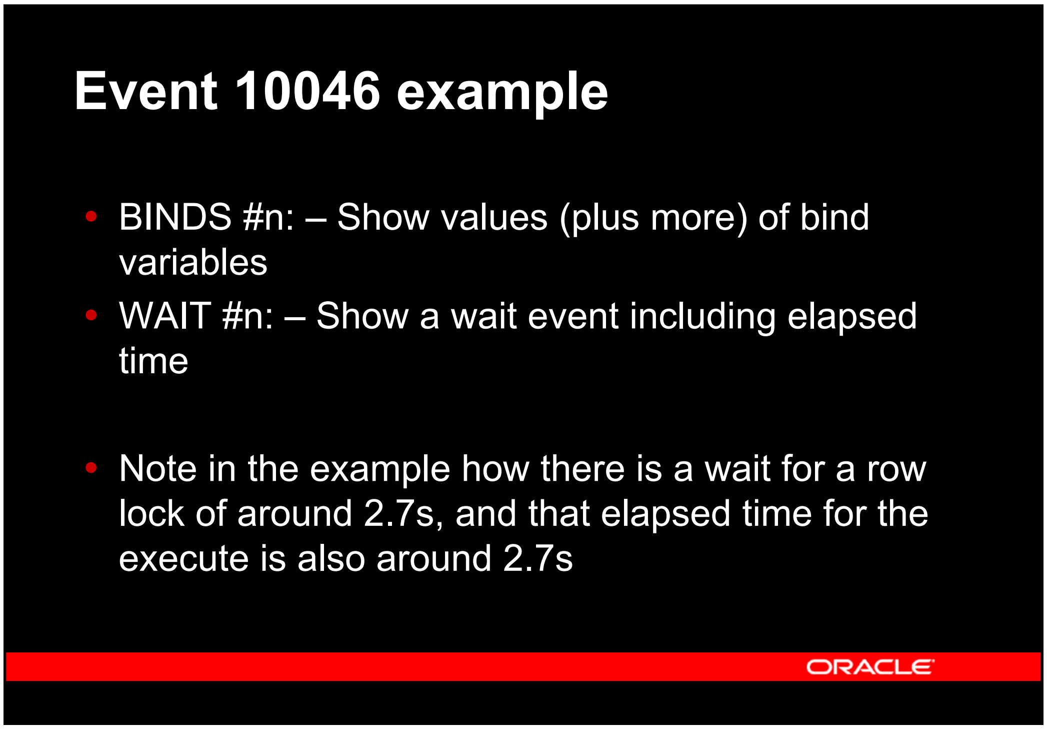
```
alter session set events
‘10046 trace name context forever, level NN’
```

- NN=1: like setting sql_trace to true
- NN=4: Trace all events
- NN=8: Trace bind variable contents
- NN=12: Trace both
- NN=0: turn off, like setting sql_trace to false



Event 10046 example

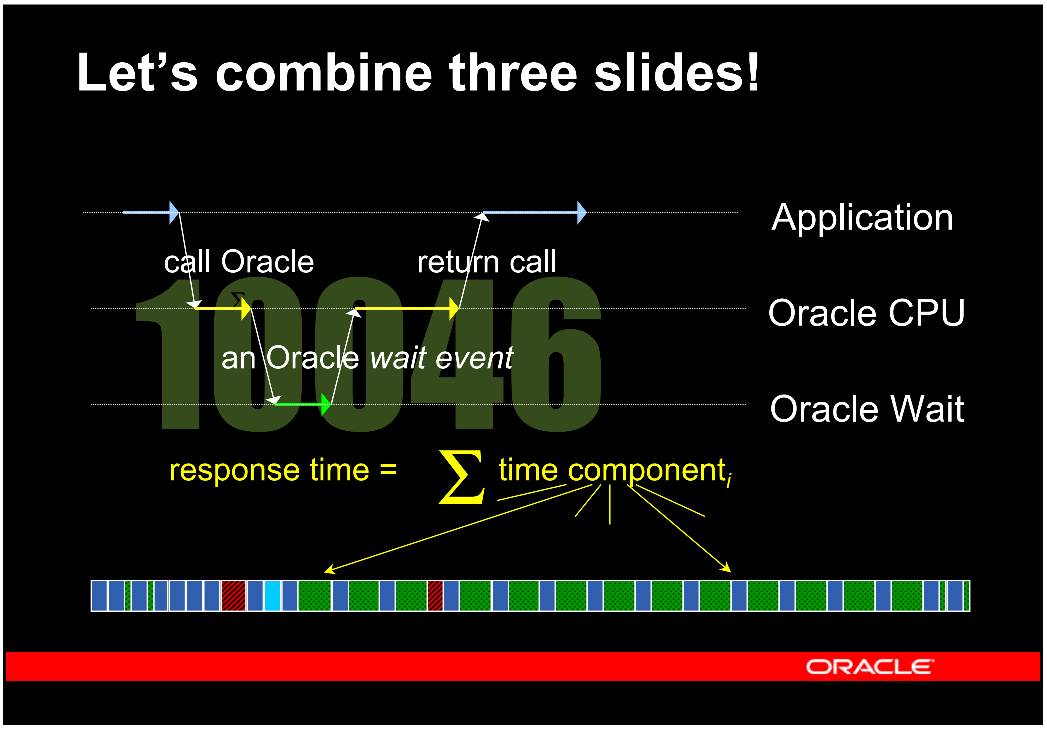
```
PARSING IN CURSOR #3 len=33 dep=0 uid=21 oct=6 lid=21
  update rac1 set b=:b1 where a=:b2
PARSE #3:c=0,e=186,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,
BINDS #3:
 Bind#0
  value=1177835187
 Bind#1
  value=23
WAIT #3: nam='enq: TX - row lock contention' ela= 2776415
EXEC #3:c=0,e=2777517,p=0,cr=2,cu=3,mis=0,r=1,dep=0,og=1,
```



Event 10046 example

- BINDS #n: – Show values (plus more) of bind variables
- WAIT #n: – Show a wait event including elapsed time

- Note in the example how there is a wait for a row lock of around 2.7s, and that elapsed time for the execute is also around 2.7s



Let’s combine three slides!

Application

call Oracle

return call

10046

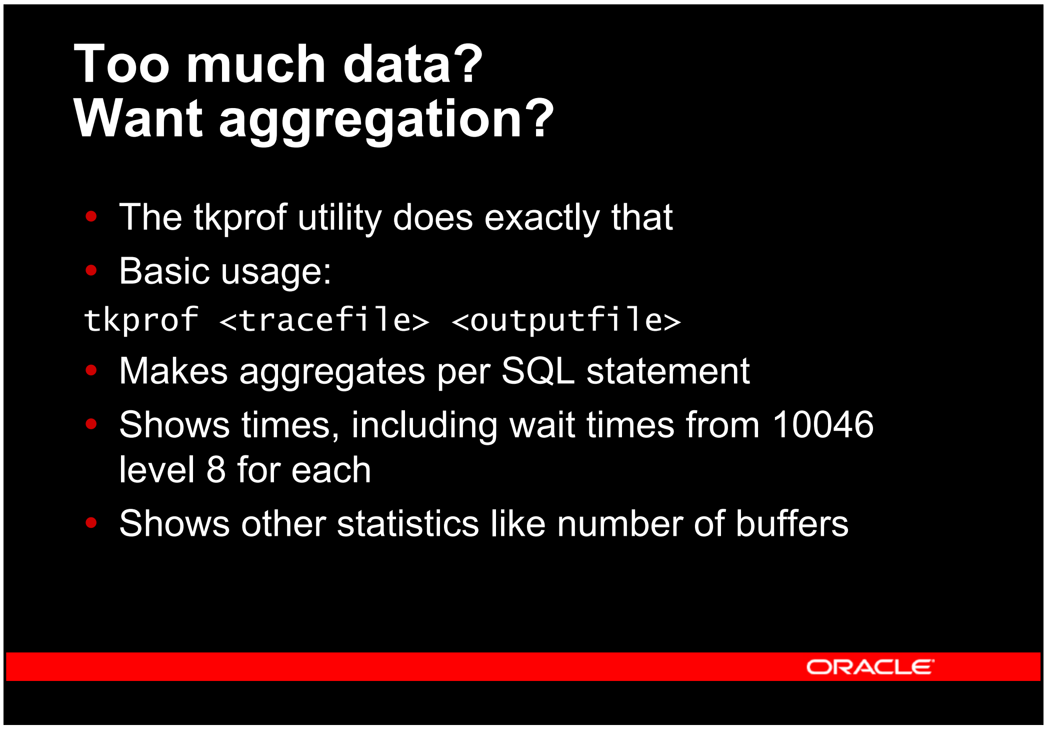
Oracle CPU

an Oracle *wait event*

Oracle Wait

response time = $\sum$ time component$_i$

ORACLE



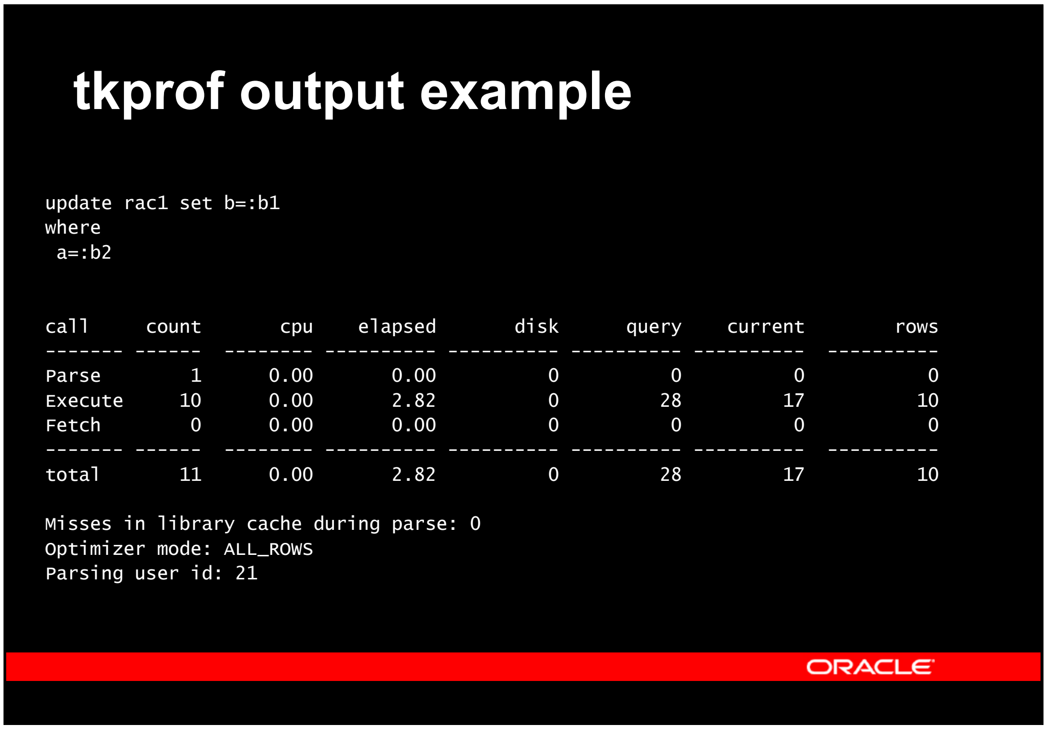
Too much data?
Want aggregation?

- The tkprof utility does exactly that
- Basic usage:

```
tkprof <tracefile> <outputfile>
```

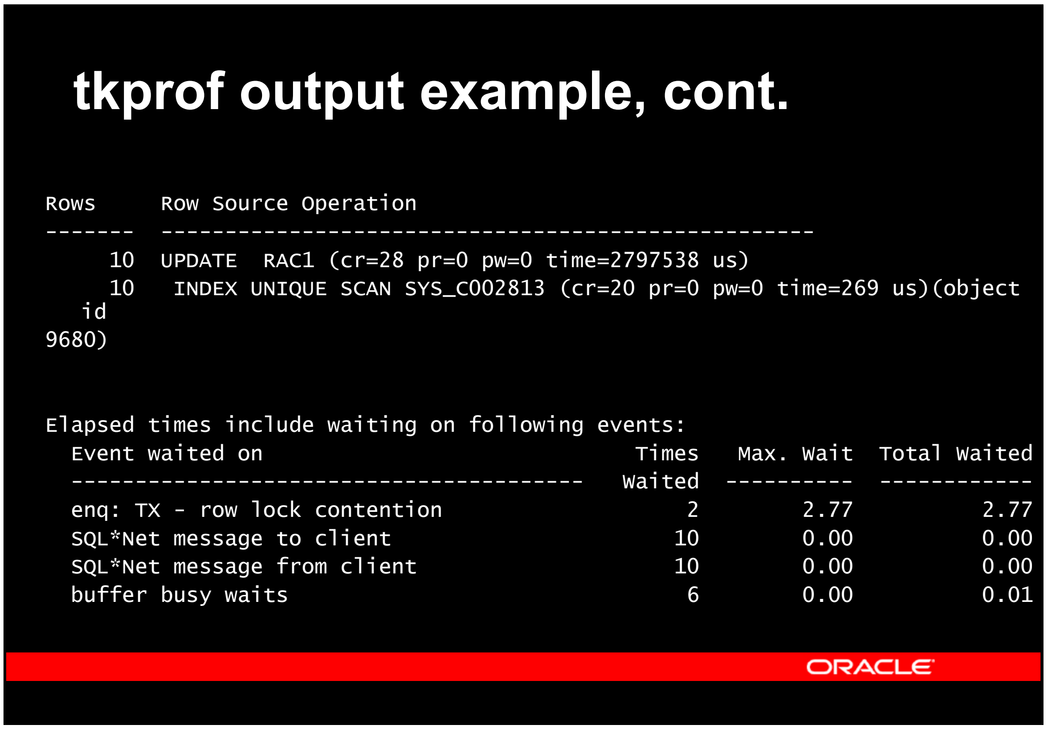
- Makes aggregates per SQL statement
- Shows times, including wait times from 10046 level 8 for each
- Shows other statistics like number of buffers

ORACLE



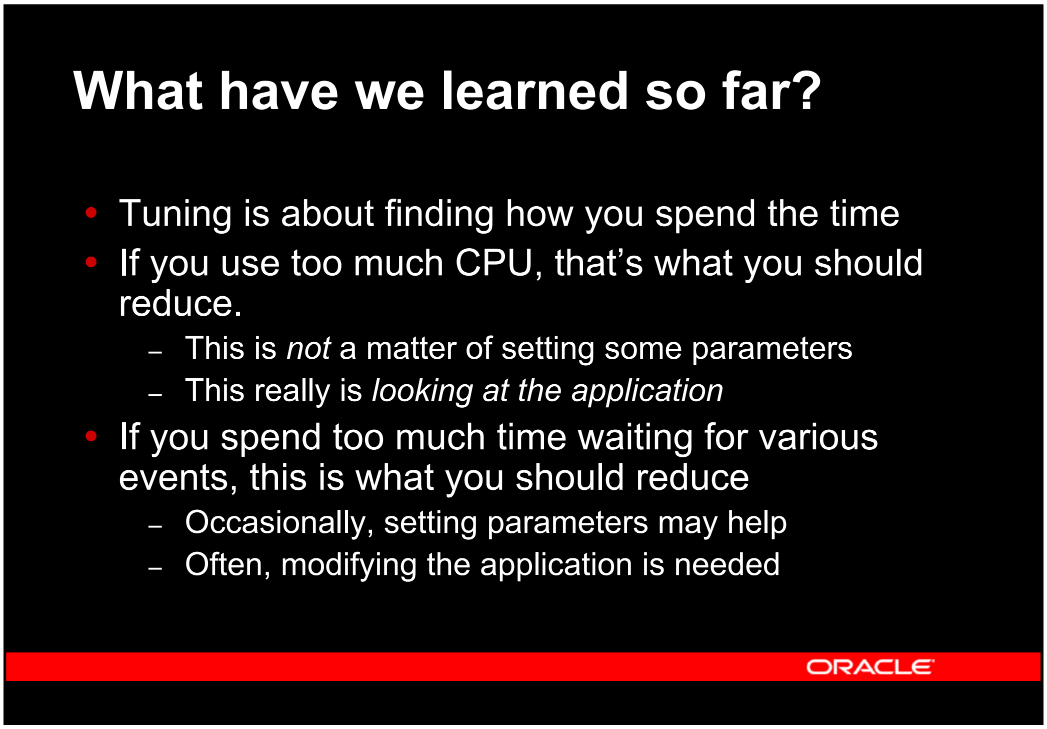
tkprof output example

```
update rac1 set b=:b1
where
 a=:b2


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute     10      0.00       2.82          0         28         17          10
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       11      0.00       2.82          0         28         17          10

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 21
```



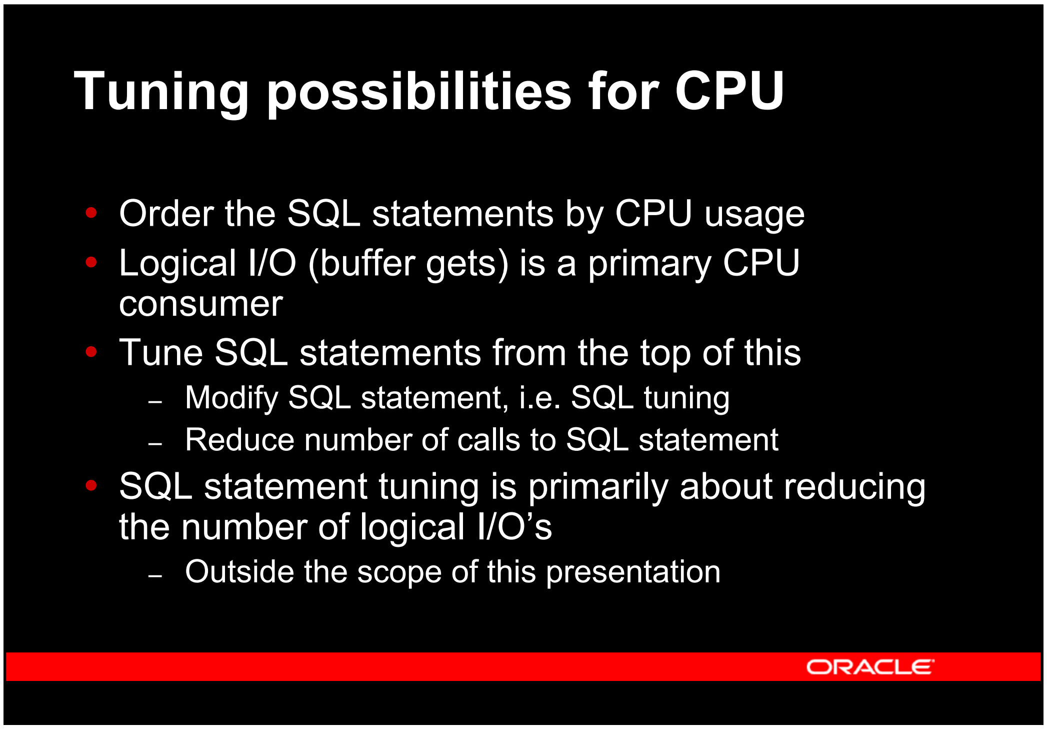
tkprof output example, cont.

```
Rows     Row Source Operation
-------  ---------------------------------------------------
     10  UPDATE  RAC1 (cr=28 pr=0 pw=0 time=2797538 us)
     10   INDEX UNIQUE SCAN SYS_C002813 (cr=20 pr=0 pw=0 time=269 us)(object
   id
9680)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  enq: TX - row lock contention                   2        2.77          2.77
  SQL*Net message to client                      10        0.00          0.00
  SQL*Net message from client                    10        0.00          0.00
  buffer busy waits                               6        0.00          0.01
```



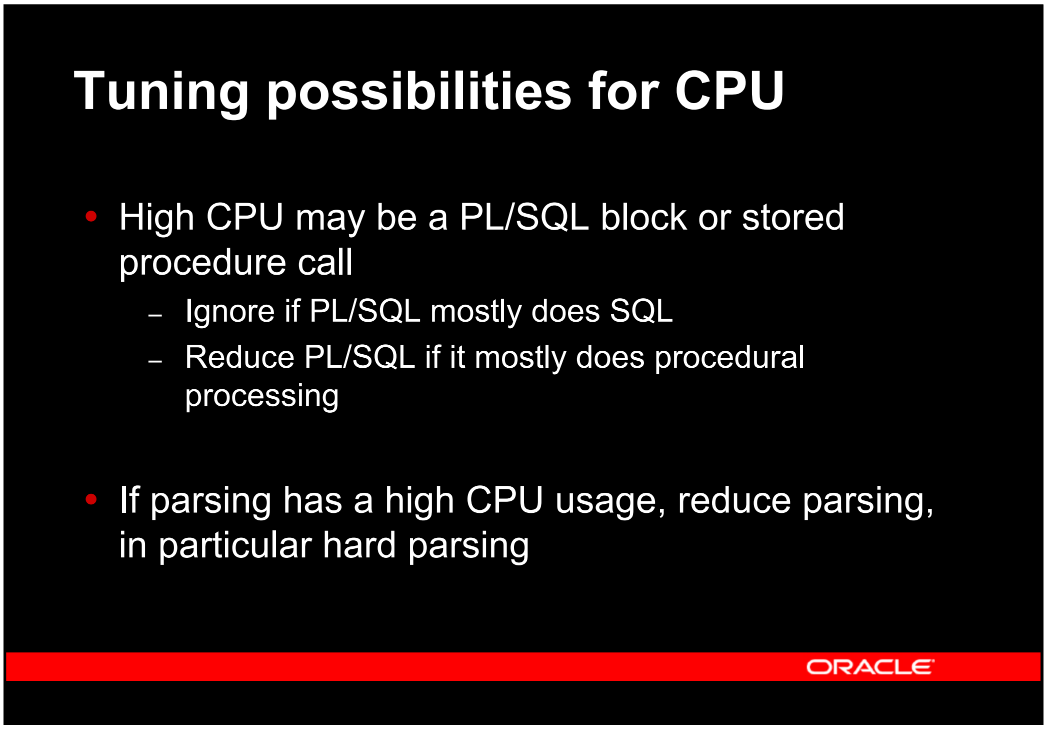
What have we learned so far?

- Tuning is about finding how you spend the time
- If you use too much CPU, that’s what you should reduce.
  - This is *not* a matter of setting some parameters
  - This really is *looking at the application*
- If you spend too much time waiting for various events, this is what you should reduce
  - Occasionally, setting parameters may help
  - Often, modifying the application is needed



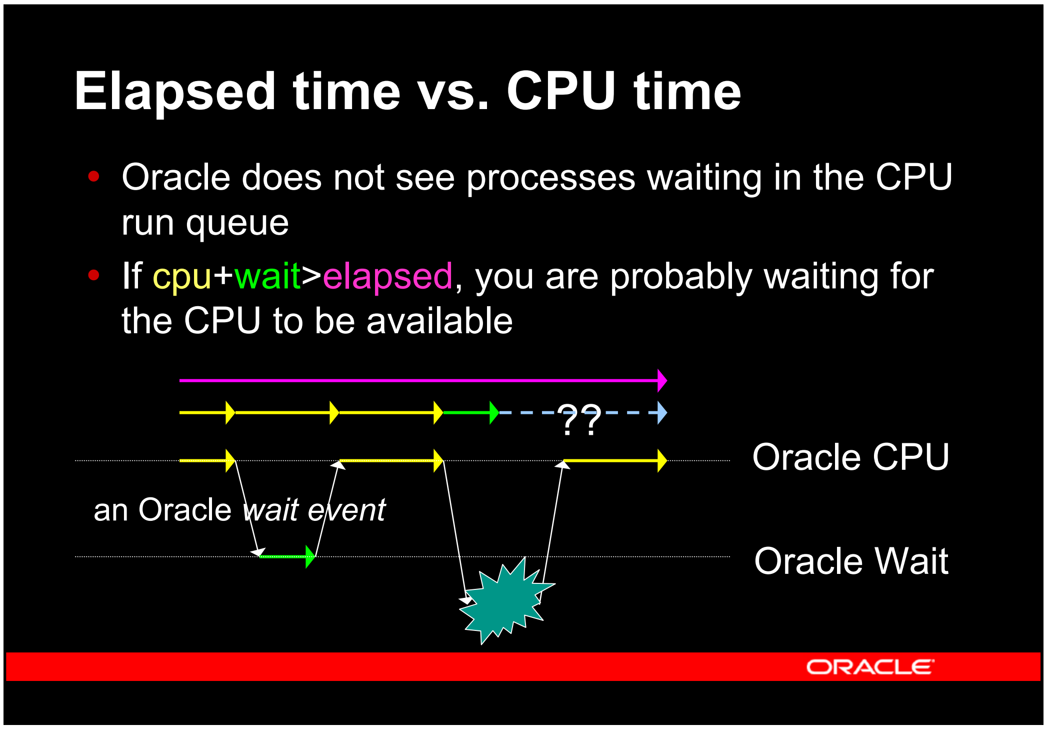
Tuning possibilities for CPU

- Order the SQL statements by CPU usage
- Logical I/O (buffer gets) is a primary CPU consumer
- Tune SQL statements from the top of this
  - Modify SQL statement, i.e. SQL tuning
  - Reduce number of calls to SQL statement
- SQL statement tuning is primarily about reducing the number of logical I/O’s
  - Outside the scope of this presentation



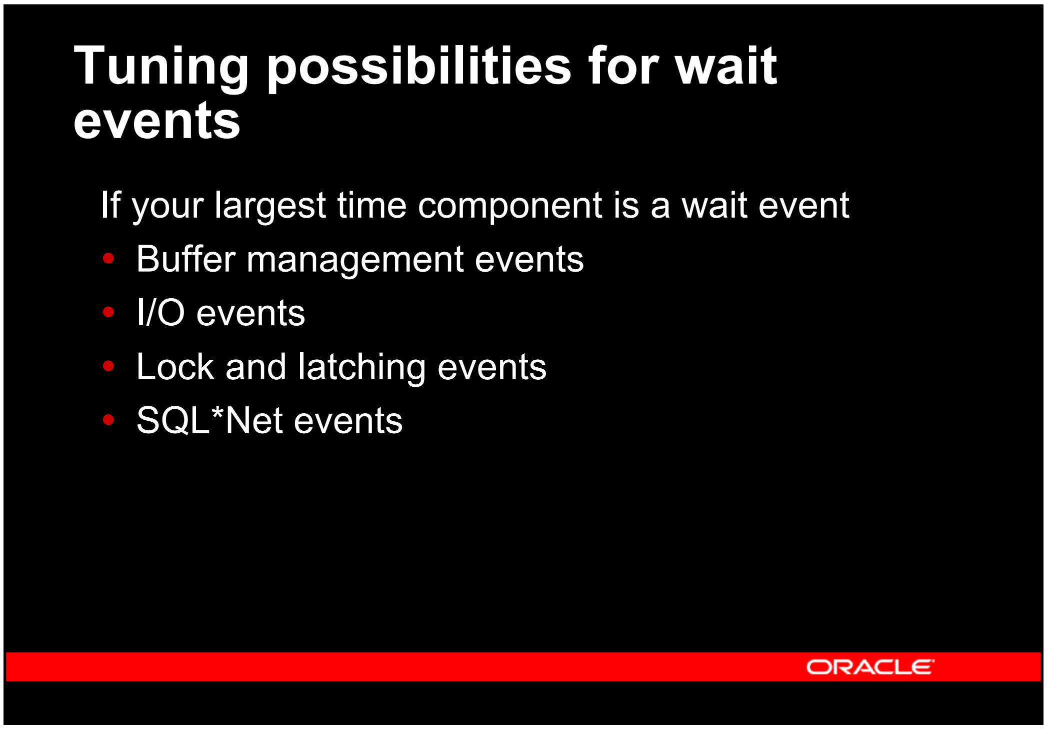
Tuning possibilities for CPU

- High CPU may be a PL/SQL block or stored procedure call
  - Ignore if PL/SQL mostly does SQL
  - Reduce PL/SQL if it mostly does procedural processing
- If parsing has a high CPU usage, reduce parsing, in particular hard parsing



Elapsed time vs. CPU time

- Oracle does not see processes waiting in the CPU run queue
- If cpu+wait>elapsed, you are probably waiting for the CPU to be available

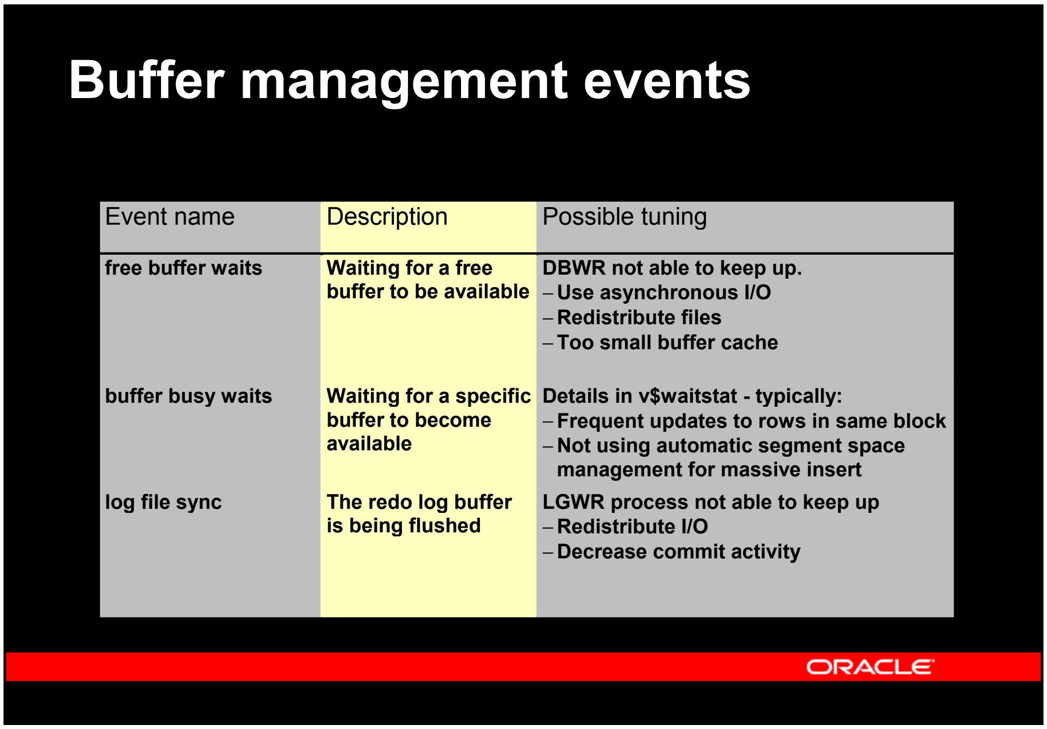


Tuning possibilities for wait events

If your largest time component is a wait event

- Buffer management events
- I/O events
- Lock and latching events
- SQL*Net events

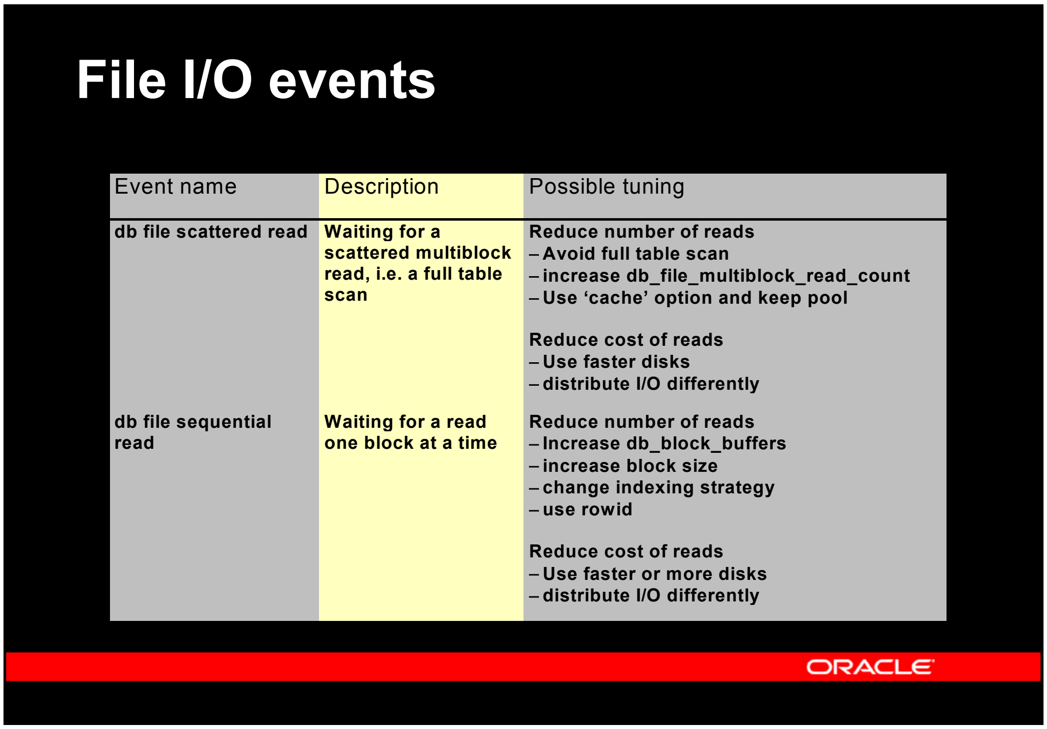
ORACLE



Buffer management events

Event name	Description	Possible tuning
free buffer waits	**Waiting for a free buffer to be available**	**DBWR not able to keep up.** **– Use asynchronous I/O** **– Redistribute files** **– Too small buffer cache**
buffer busy waits	**Waiting for a specific buffer to become available**	**Details in v$waitstat - typically:** **– Frequent updates to rows in same block** **– Not using automatic segment space management for massive insert**
log file sync	**The redo log buffer is being flushed**	**LGWR process not able to keep up** **– Redistribute I/O** **– Decrease commit activity**

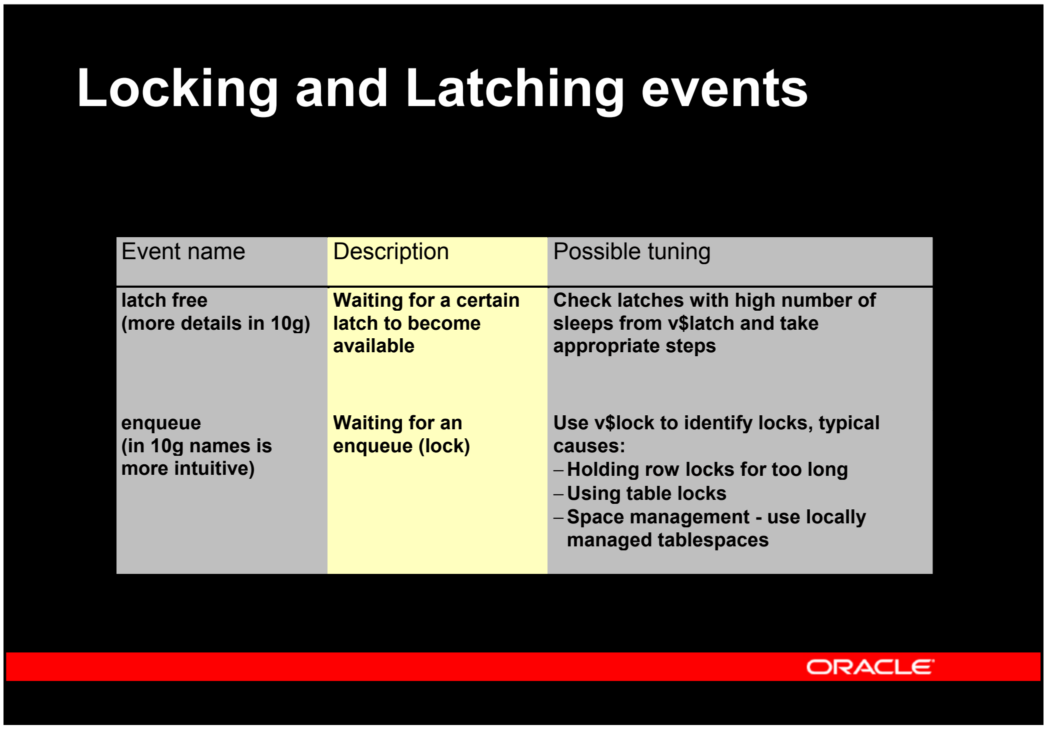
ORACLE



File I/O events

Event name	Description	Possible tuning
db file scattered read	Waiting for a scattered multiblock read, i.e. a full table scan	Reduce number of reads – Avoid full table scan – increase db_file_multiblock_read_count – Use 'cache' option and keep pool Reduce cost of reads – Use faster disks – distribute I/O differently
db file sequential read	Waiting for a read one block at a time	Reduce number of reads – Increase db_block_buffers – increase block size – change indexing strategy – use rowid Reduce cost of reads – Use faster or more disks – distribute I/O differently

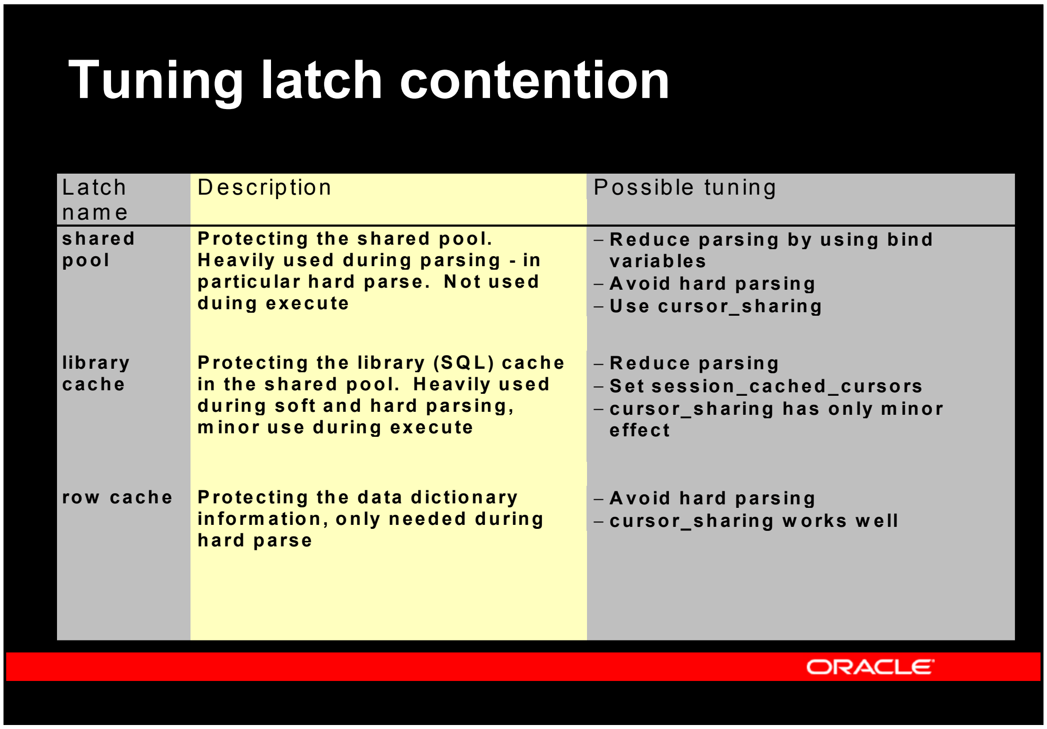
ORACLE



Locking and Latching events

Event name	Description	Possible tuning
latch free (more details in 10g)	Waiting for a certain latch to become available	Check latches with high number of sleeps from v$latch and take appropriate steps
enqueue (in 10g names is more intuitive)	Waiting for an enqueue (lock)	Use v$lock to identify locks, typical causes: – Holding row locks for too long – Using table locks – Space management - use locally managed tablespaces

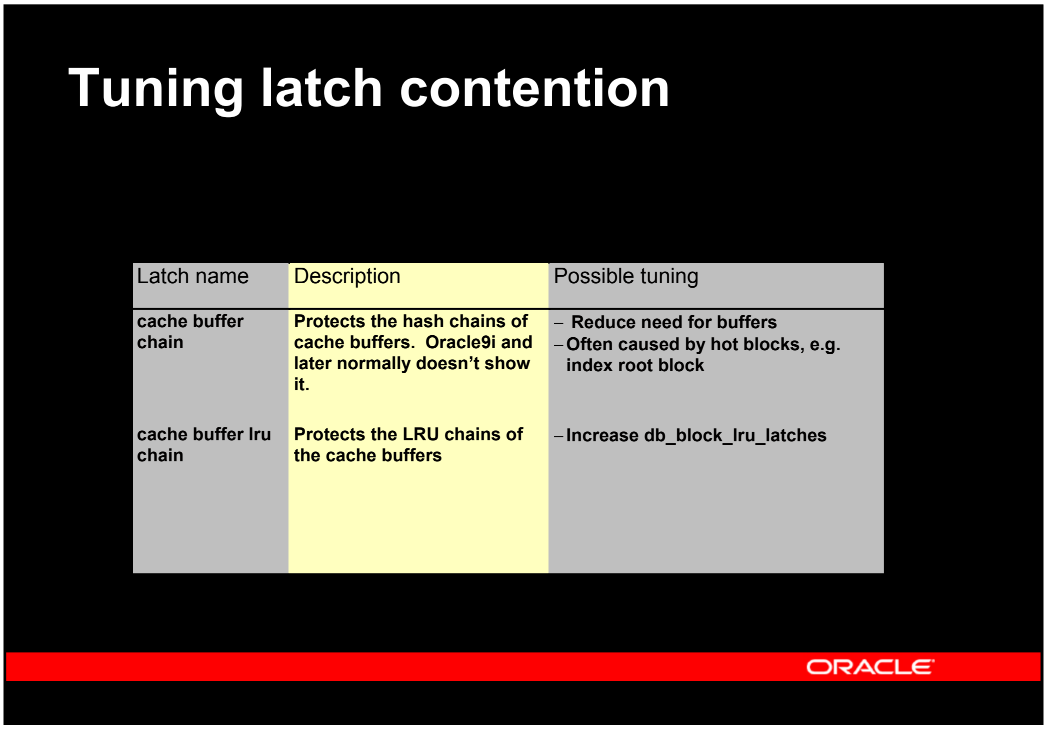
ORACLE



Tuning latch contention

Latch name	Description	Possible tuning
shared pool	Protecting the shared pool. Heavily used during parsing - in particular hard parse. Not used duing execute	– Reduce parsing by using bind variables – Avoid hard parsing – Use cursor_sharing
library cache	Protecting the library (SQL) cache in the shared pool. Heavily used during soft and hard parsing, minor use during execute	– Reduce parsing – Set session_cached_cursors – cursor_sharing has only minor effect
row cache	Protecting the data dictionary information, only needed during hard parse	– Avoid hard parsing – cursor_sharing works well

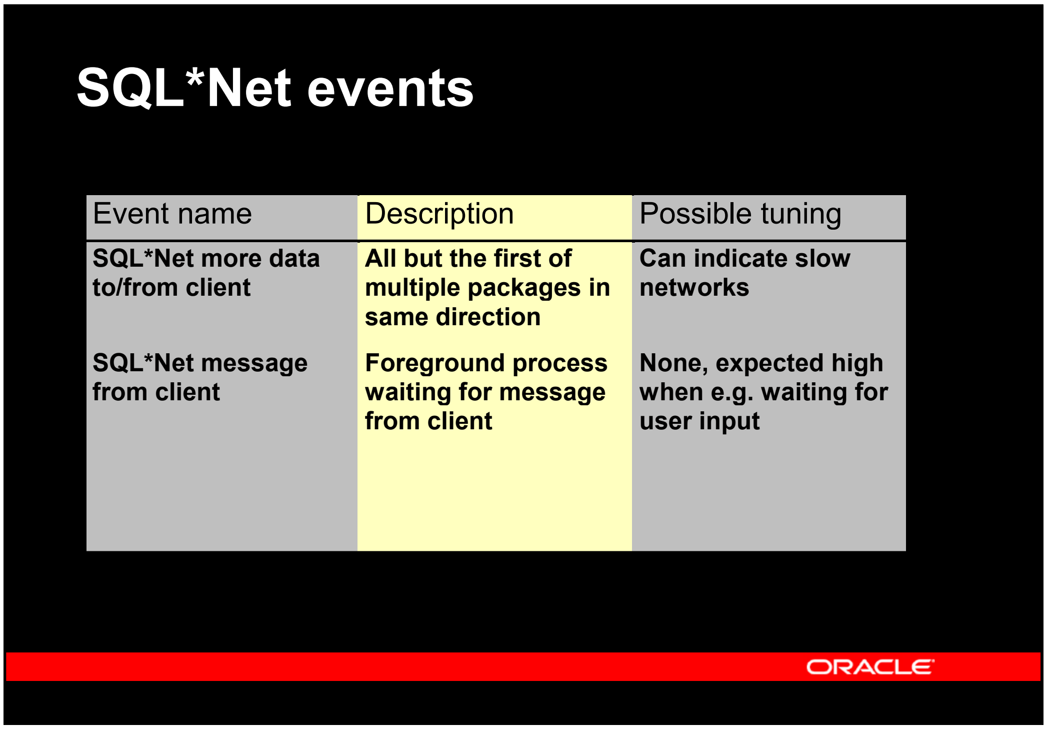
ORACLE



Tuning latch contention

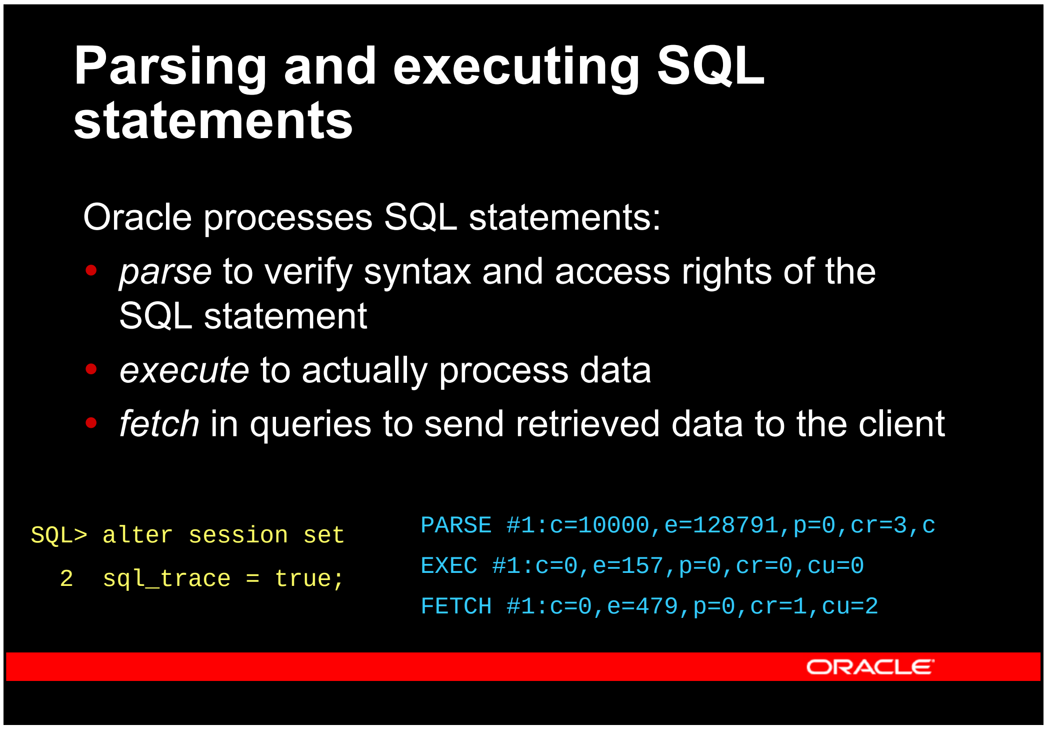
Latch name	Description	Possible tuning
cache buffer chain	Protects the hash chains of cache buffers. Oracle9i and later normally doesn't show it.	– Reduce need for buffers – Often caused by hot blocks, e.g. index root block
cache buffer lru chain	Protects the LRU chains of the cache buffers	– Increase db_block_lru_latches

ORACLE



SQL*Net events

Event name	Description	Possible tuning
SQL*Net more data to/from client	**All but the first of multiple packages in same direction**	**Can indicate slow networks**
SQL*Net message from client	**Foreground process waiting for message from client**	**None, expected high when e.g. waiting for user input**



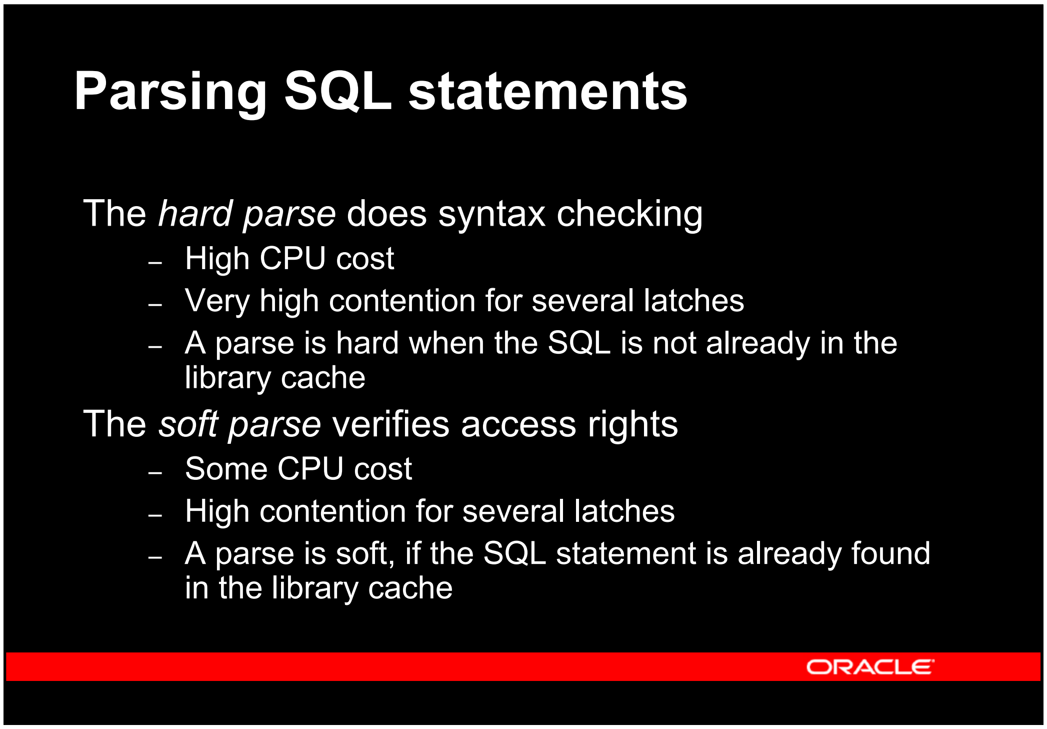
Parsing and executing SQL statements

Oracle processes SQL statements:

- *parse* to verify syntax and access rights of the SQL statement
- *execute* to actually process data
- *fetch* in queries to send retrieved data to the client

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

```
PARSE #1:c=10000,e=128791,p=0,cr=3,c
EXEC #1:c=0,e=157,p=0,cr=0,cu=0
FETCH #1:c=0,e=479,p=0,cr=1,cu=2
```



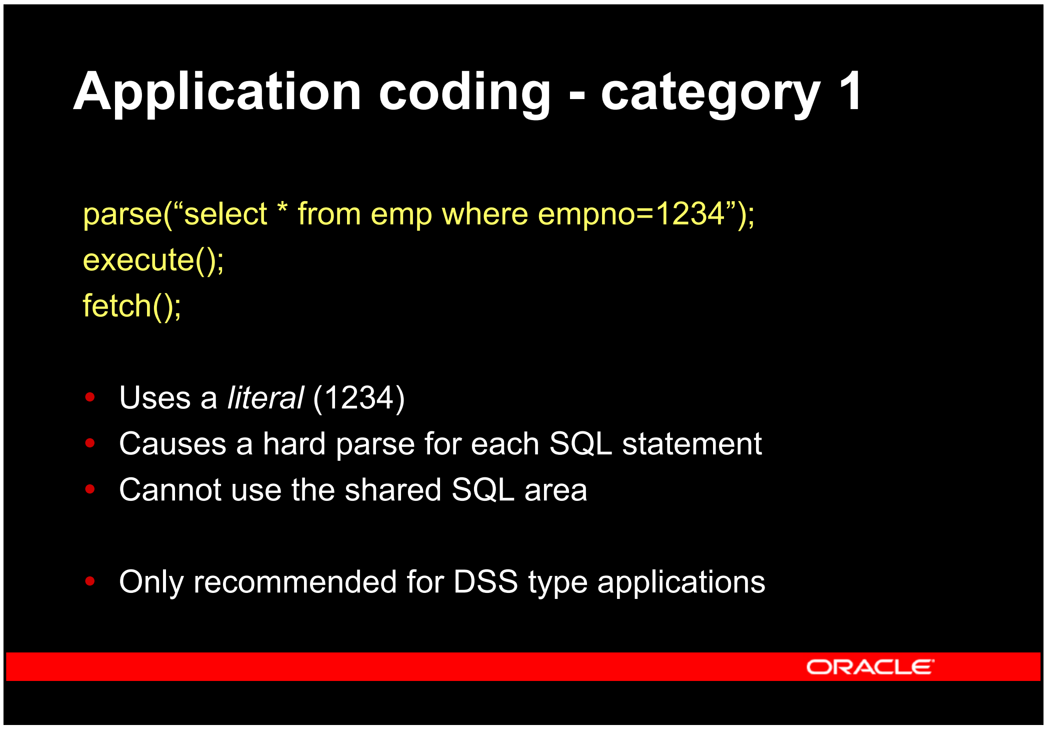
Parsing SQL statements

The *hard parse* does syntax checking

- High CPU cost
- Very high contention for several latches
- A parse is hard when the SQL is not already in the library cache

The *soft parse* verifies access rights

- Some CPU cost
- High contention for several latches
- A parse is soft, if the SQL statement is already found in the library cache

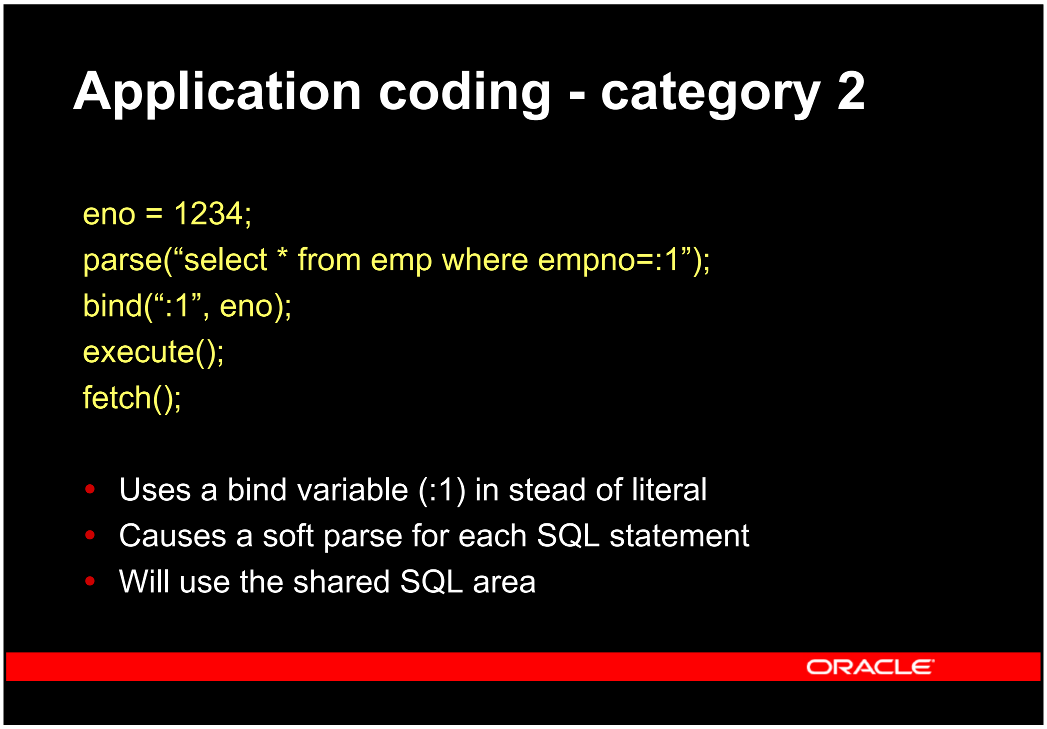


Application coding - category 1

```
parse(“select * from emp where empno=1234”);
execute();
fetch();
```

- Uses a *literal* (1234)
- Causes a hard parse for each SQL statement
- Cannot use the shared SQL area

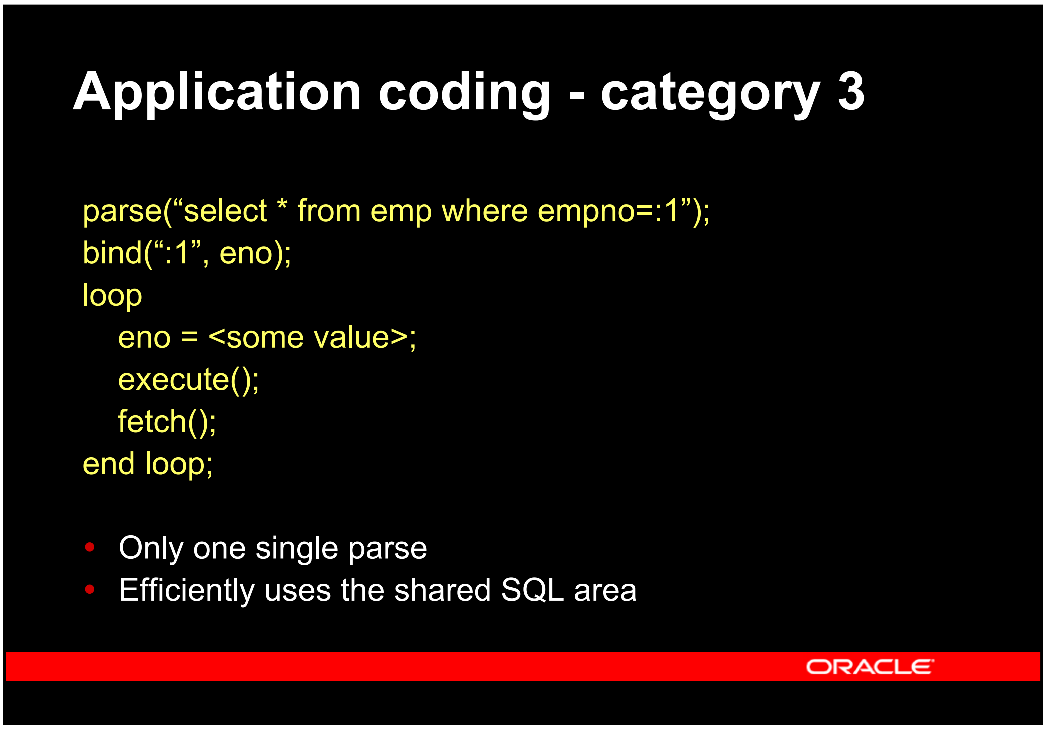
- Only recommended for DSS type applications



Application coding - category 2

```
eno = 1234;
parse(“select * from emp where empno=:1”);
bind(“:1”, eno);
execute();
fetch();
```

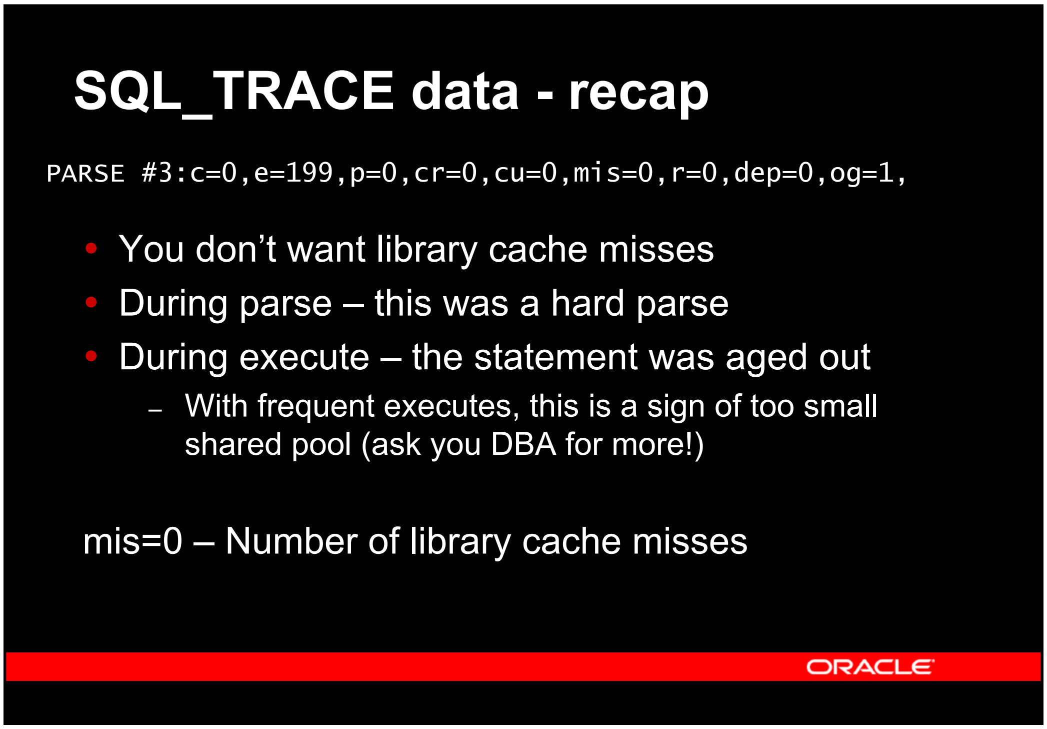
- Uses a bind variable (:1) in stead of literal
- Causes a soft parse for each SQL statement
- Will use the shared SQL area



Application coding - category 3

```
parse(“select * from emp where empno=:1”);
bind(“:1”, eno);
loop
   eno = <some value>;
   execute();
   fetch();
end loop;
```

- Only one single parse
- Efficiently uses the shared SQL area

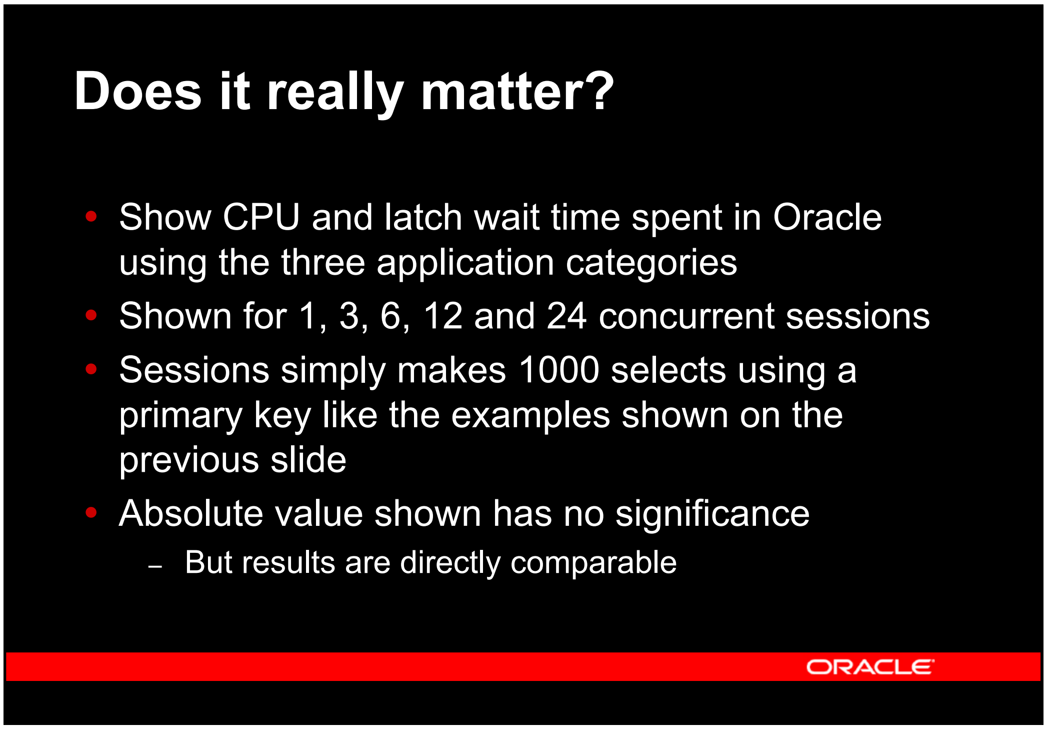


SQL_TRACE data - recap

```
PARSE #3:c=0,e=199,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,
```

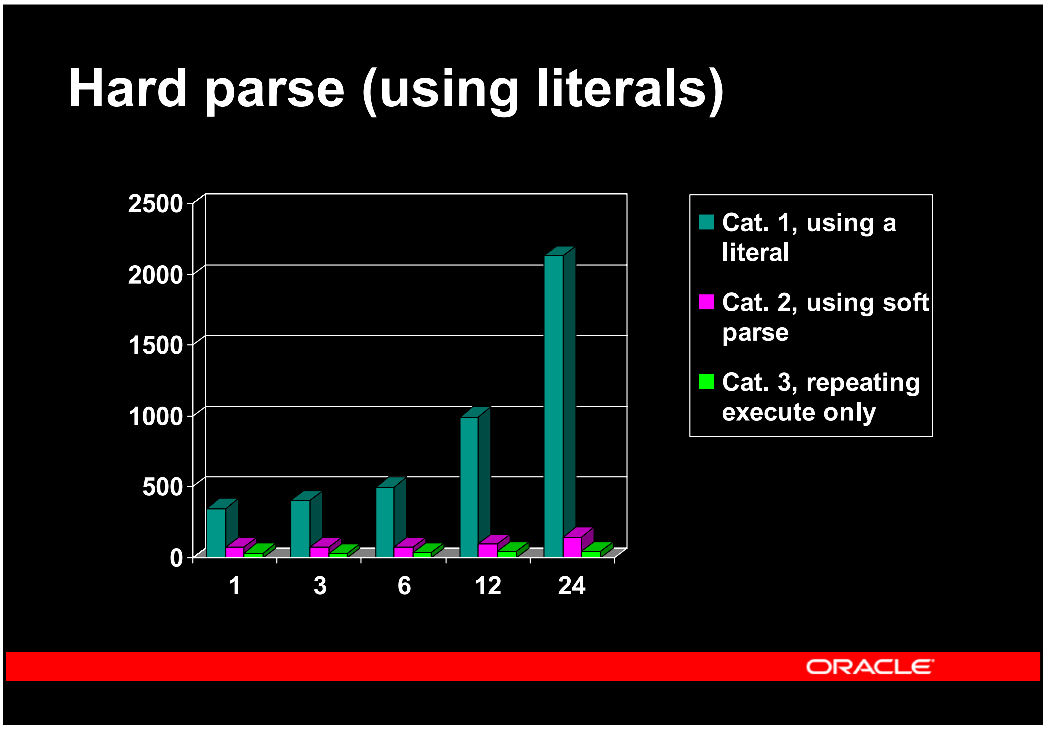
- You don't want library cache misses
- During parse – this was a hard parse
- During execute – the statement was aged out
  - With frequent executes, this is a sign of too small shared pool (ask you DBA for more!)

mis=0 – Number of library cache misses

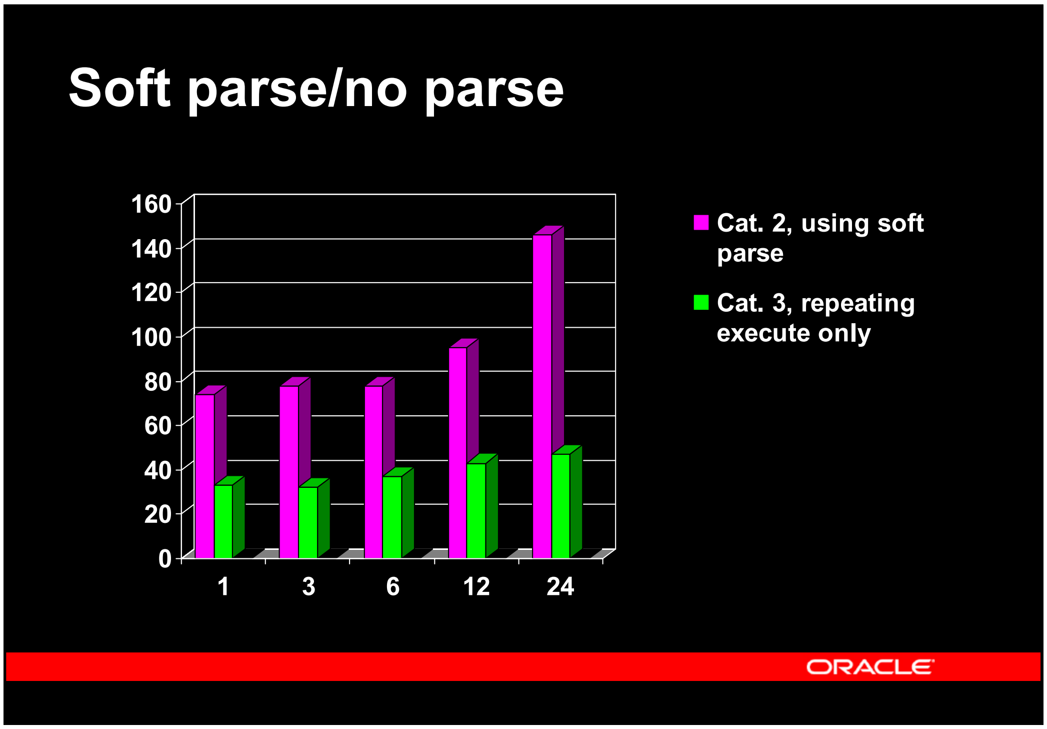


Does it really matter?

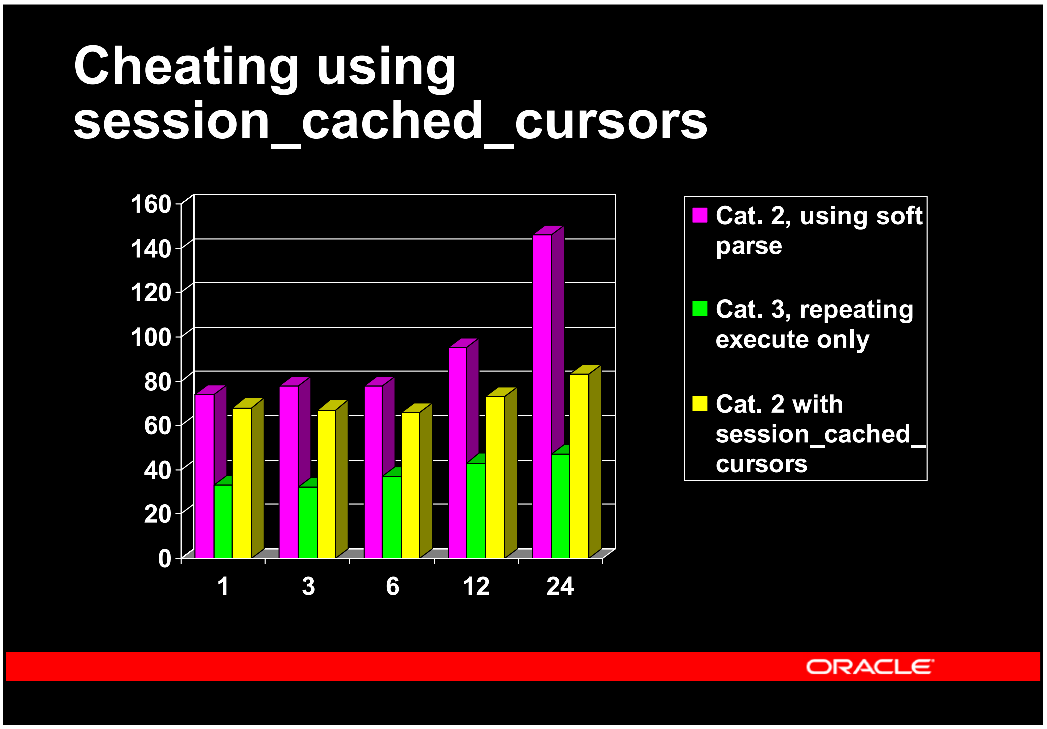
- Show CPU and latch wait time spent in Oracle using the three application categories
- Shown for 1, 3, 6, 12 and 24 concurrent sessions
- Sessions simply makes 1000 selects using a primary key like the examples shown on the previous slide
- Absolute value shown has no significance
  - But results are directly comparable



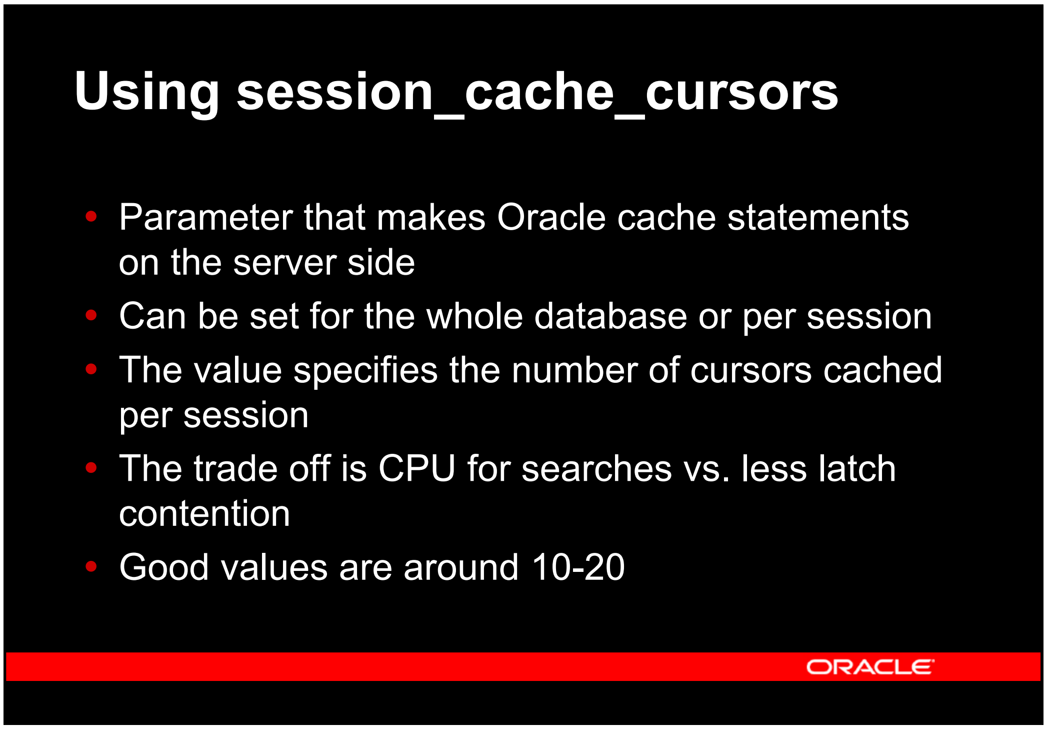
Hard parse (using literals)



Soft parse/no parse

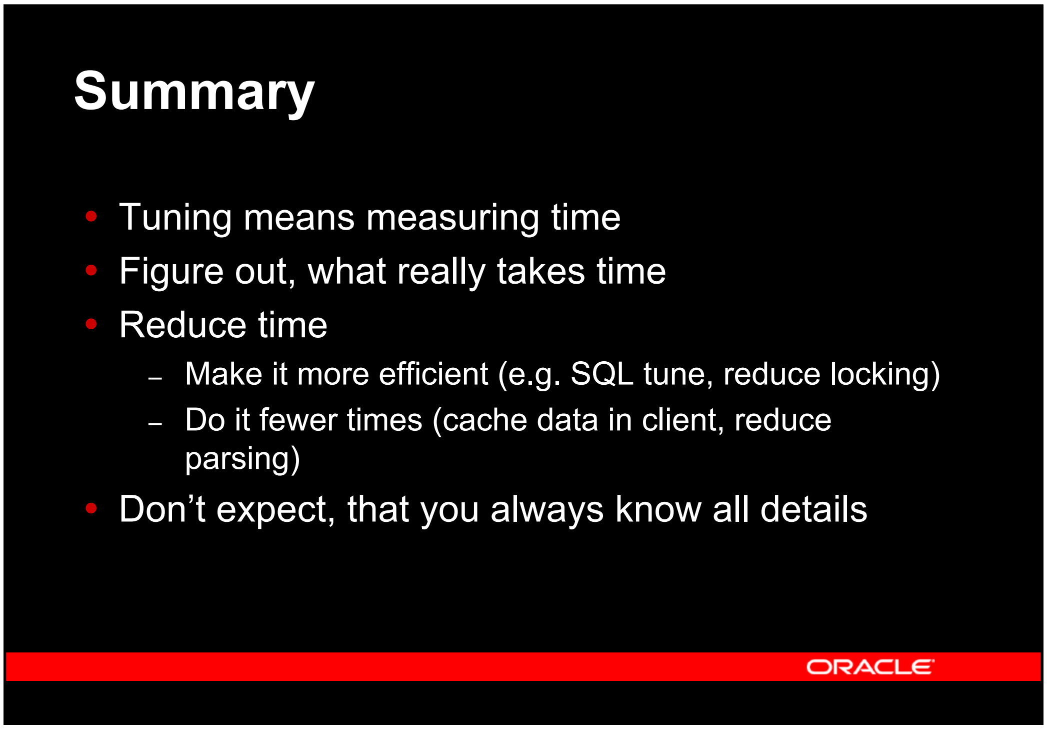


Cheating using session_cached_cursors



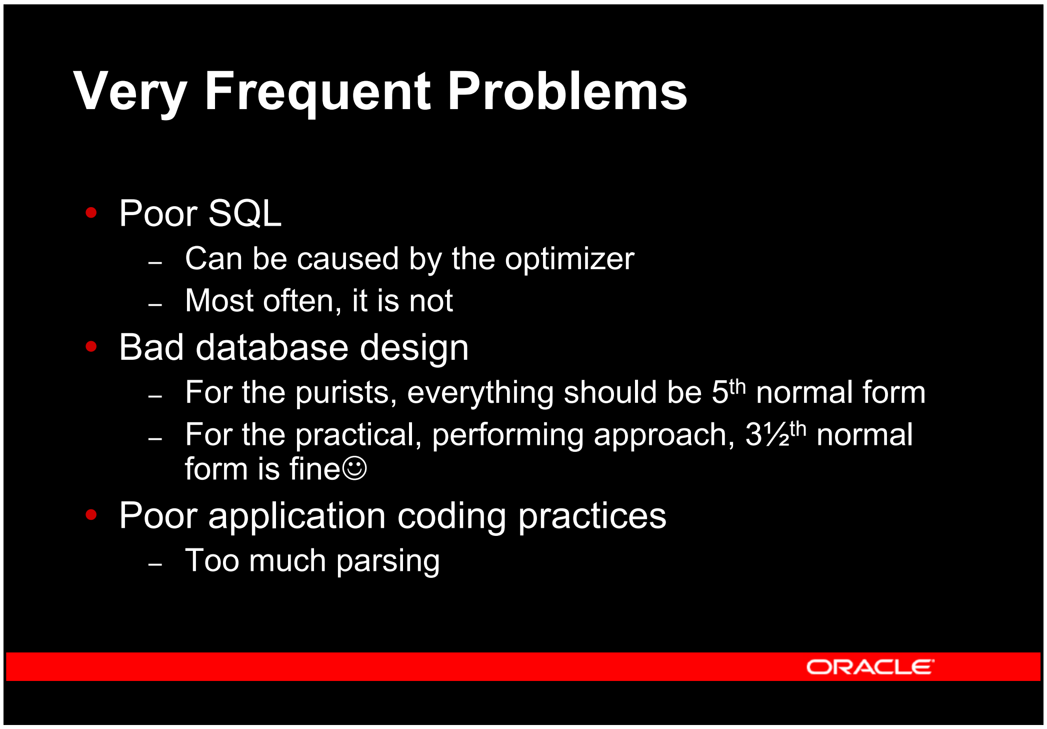
Using session_cache_cursors

- Parameter that makes Oracle cache statements on the server side
- Can be set for the whole database or per session
- The value specifies the number of cursors cached per session
- The trade off is CPU for searches vs. less latch contention
- Good values are around 10-20



Summary

- Tuning means measuring time
- Figure out, what really takes time
- Reduce time
  - Make it more efficient (e.g. SQL tune, reduce locking)
  - Do it fewer times (cache data in client, reduce parsing)
- Don’t expect, that you always know all details



Very Frequent Problems

- Poor SQL
  - Can be caused by the optimizer
  - Most often, it is not
- Bad database design
  - For the purists, everything should be 5th normal form
  - For the practical, performing approach, 3½th normal form is fine☺
- Poor application coding practices
  - Too much parsing

QUESTIONS
ANSWERS