

The Tools I Use

What we'll be doing...

- My 3 best friends
 - Autotrace
 - Tkprof
 - statspack
- Their friends
 - dbms_profiler
 - Runstats
 - And even jdev
- Instrument your code

Break Glass in case of Emergency

- Triage
 - In most cases you have no prior information. You are starting from scratch
 - Get a statspack
 - Isolate an application
 - Trace it
- If you implement some of the ideas I'll talk about
 - You won't have to triage (you can predict)
 - When you need to isolate where the issue is – you can
 - Finding the problem is almost always harder than fixing it! (time entry example)

My three best friends

- Autotrace and DBMS_XPLAN
 - Too easy
 - Explain plan
 - The *statistics*
 - ***autotrace.sql***

My three best friends

- ✓ We have a big customer table which is referenced by many other tables.
- ✓ The table's primary key is NUMBER(12). However the foreign key columns of the some of the tables which is referencing this customer table were defined as NUMBER. (Seen as NUMBER(38) in data dictionary).
- ✓ This datatype difference causes a data conversion during the checking of the foreign keys and thus a performance loss. (*invalid conclusion based on guessing – no hard facts to back it up*)
- ✓ In order to prevent it, we tried to alter the NUMBER columns to the same type of our customer table's primary key - NUMBER(12).
- ✓ Since Oracle doesn't allow us to do this unless the column values are NULL, we had to insert them with their rowids to a temporary table, setting them to NULL, altering the column to NUMBER(12), and populating the original values from the temporary table again. (*doing a TON of work and they will see no gain from this – other then tons of IO*)
- ✓ This is a very slow process. So that we can't even see its end after 8 hours of working and we had to stop it. My question is, could you please offer us a faster way of doing this operation? (*yes we can – stop doing it!*)
- ✓ The total size of the tables we have to alter is about 220 million rows, we are running all DML in parallel and nologging mode, we are creating the temporary table by create as select

- Review tkprof.sql and tkprof.prf

My three best friends

- Statspack

- Keep a history

- *I've been asked to look at a site who have just carried out a DB upgrade. The feeling is that there has been some performance degradation since the upgrade. The performance hit has not been quantified yet but is 'felt throughout the day'.*

- Use it in sickness and in health

- 15-30 minute windows between snaps at most

- Averaging out over 8 hours is meaningless

- Is an hourly observation useful?

- Don't try to remove every last wait

- *When looking through the statspack report, I found that events, "direct path read" and "direct path write", are always on the top 2 of "Top 5 Wait Events" section as follows. They had 2.92 seconds of waits on it– over a 30minute window!*
 - Watch for the “so what” waits – control file parallel write

My three best friends

- So, you have a 20 page statspack report.
- And you want to “tune” with it
 - Make sure it is 15-30 minutes long
 - Make sure timed statistics were on

STATSPACK report for

DB Name	DB Id	Instance	Inst Num	Release	Cluster	Host
ORA9I	2272536868	ora9i	1	9.2.0.1.0	NO	aria

	Snap Id	Snap Time	Sessions	Curs/Sess	Comment
Begin Snap:	1	30-Dec-02 09:58:58	67,254	3.0	
End Snap:	2	30-Dec-02 10:14:52	67,260	3.0	
Elapsed:		15.90 (mins)			

Cache Sizes (end)

~~~~~

|                   |      |                 |      |
|-------------------|------|-----------------|------|
| Buffer Cache:     | 96M  | Std Block Size: | 8K   |
| Shared Pool Size: | 112M | Log Buffer:     | 512K |

# My three best friends

- Check hard parses, want almost none
- Executes & TPS is an indicator of the “load” on the system
- Others are useful for reference

## Load Profile

~~~~~

	Per Second	Per Transaction
	-----	-----
Redo size:	75,733.92	20,737.70
Logical reads:	1,535.11	420.35
Block changes:	449.56	123.10
Physical reads:	562.99	154.16
Physical writes:	62.53	17.12
User calls:	8.04	2.20
Parses:	56.43	15.45
Hard parses:	0.38	0.10
Sorts:	11.63	3.19
Logons:	0.30	0.08
Executes:	94.21	25.80
Transactions:	3.65	

% Blocks changed per Read:	29.29	Recursive Call %:	97.14
Rollback per transaction %:	9.41	Rows per Sort:	752.04

My three best friends

- The ratios – Library hit, Soft parse are the ones I zero in on
- Buffer hit – not so much
- Execute to parse – depends on system type
 - Negative = bad
 - 20-40% not unreasonable for web based (stateless)
 - Client/server should be much much higher.

Instance Efficiency Percentages (Target 100%)

```
~~~~~  
          Buffer Nowait %: 100.00          Redo NoWait %: 100.00  
          Buffer Hit %: 94.94          In-memory Sort %: 98.50  
          Library Hit %: 99.81          Soft Parse %: 99.33  
          Execute to Parse %: 40.10          Latch Hit %: 99.91  
Parse CPU to Parse Elapsed %: 86.12          % Non-Parse CPU: 94.69
```

```
Shared Pool Statistics          Begin          End  
-----          -----  
          Memory Usage %: 86.95          85.16  
          % SQL with executions>1: 66.38          67.40  
          % Memory for SQL w/exec>1: 67.28          69.68
```

Ratios

- Definition:
 - (1) a mathematical device used to hide and obscure information from vision
 - (2) not a goal

Instance Efficiency Percentages (Target 100%)

~~~~~

|                               |       |                   |        |
|-------------------------------|-------|-------------------|--------|
| Buffer Nowait %:              | 99.99 | Redo NoWait %:    | 99.99  |
| Buffer Hit %:                 | 93.16 | In-memory Sort %: | 100.00 |
| Library Hit %:                | 99.61 | Soft Parse %:     | 99.59  |
| Execute to Parse %:           | 11.79 | Latch Hit %:      | 99.98  |
| Parse CPU to Parse Elapsed %: | 11.36 | % Non-Parse CPU:  | 97.69  |

how can i get rid of low values of execute to parse and Parse CPU to Parse Elapsed %.

# Ratios

- Remember 1 second of CPU, 2 seconds of Elapsed would be a 50% parse cpu/ela ratio!
- Suppose In  $\frac{1}{2}$  hour you have...
  - 500 elapsed seconds of parse time
  - 50 seconds CPU seconds of parse time
  - That is a 10% ratio
  - But, you had 1,000 concurrent users
  - So, they waited a sum of 450 seconds (ela-cpu)
  - Or an average of 0.45 seconds each in  $\frac{1}{2}$  hour
  - hmmm

# My three best friends

- Top 5 timed events
  - Not waits in 9iR2 and up – timed events, includes CPU time
  - This was a 4 CPU machine, I used 508 out of 3,600 CPU seconds
    - *Or did I??????????????*
  - Direct path write caught my eye
    - Direct loads
    - PDML
    - Uncached lobs
    - *Sorts to disk* ←
- The rest of the report helps me figure out the top of the report

Top 5 Timed Events

| Event                    | Waits        | Time (s)   | % Total<br>Ela Time |
|--------------------------|--------------|------------|---------------------|
| -----                    | -----        | -----      | -----               |
| CPU time                 |              | 508        | 37.39               |
| <b>direct path write</b> | <b>5,168</b> | <b>279</b> | <b>20.50</b>        |
| db file scattered read   | 38,554       | 270        | 19.85               |
| log file sync            | 2,610        | 88         | 6.50                |
| direct path read         | 2,702        | 86         | 6.29                |
| -----                    | -----        | -----      | -----               |

# Their three best friends

- DBMS\_PROFILER
  - Source code profiler for PLSQL
  - Find the low hanging fruit
  - *Dbmsprof.sql*
- Runstats
  - Simple test harness to show differences in approaches
  - *rstats.sql*
- Jdeveloper
  - Source code DEBUGGER for PLSQL
  - Interactive develop/compile/debug or...
  - Debug from afar
    - Submit web page
    - Attach debugger to mod\_plsql invoked procedure

# Instrumentation

- What the heck is this?
- Common pushback:
  - This is overhead
  - This will slow down the code
  - This is extra code I don't need
- My answer
  - Why are you curious about the v\$ tables, guess what they are
  - SQL\_TRACE?
  - The entire Events subsystem
  - Do I practice what I preach?
    - *[Trace\\_yes.html](#) [trace\\_no.html](#)*

# Instrumentation

- Databases were born to write to – every click on every web site I build comes with a builtin insert
- *For the last 24 hours, I have observed that accessing your site, clicking on 'ask question', 'read question', 'Review Question' etc., all gone slow (taking around 2-3 minutes instead of couple of seconds before). Are others facing the same problem?*
- I just went to my statistics page, generated right from my audit trail as part of my system
- Owarepl [owarepl\\_index.html](#) [ocidesc.c](#)
- A certain Apps implementation I worked on
  - *(that didn't do this)*

# Instrumentation

- Use DBMS\_APPLICATION\_INFO everywhere!
  - *appinfo.sql*
- Debug.f
  - *debugf.sql*
- Make it so your applications can all enable SQL\_TRACE! (&p\_trace=YES)
- If you use java – use the J2SE/EE industry standard logging. Help us help you
- Audit is 5 letters, not 4

# Build a Test Environment

- **Certain large institution**
  - Upgraded
  - “Seemed Slower”
  - Flipped 16 switches and didn’t have the ability unswitch them
- A certain city asked on Friday if there were any “gotchas” with upgrading – they just wanted to know for tomorrow
- Testing will
  - Benchmark it, test scale, test performance
  - Verify fixes actually work
  - Assure you that upgrade script actually works
  - Make sure major things don’t knock you out

*Questions  
and  
Answers*