

# Advanced Oracle Troubleshooting – Live Session

**Randolf Geist**

<http://oracle-randolf.blogspot.com/>  
<http://www.sqltools-plusplus.org:7676/>  
[info@sqltools-plusplus.org](mailto:info@sqltools-plusplus.org)

# Who am I


---

- Independent Consultant
- Located in Germany
- Oracle ACE
- OCP 8i, 9i, 10g
- Member of the OakTable Network  
<http://www.oaktable.net>



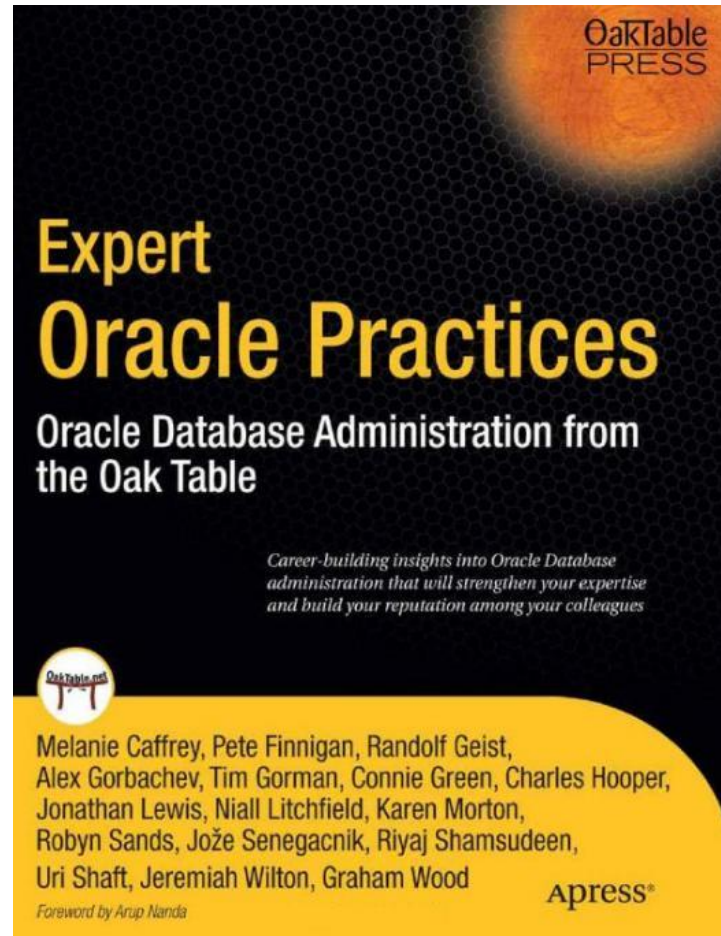
# Who am I

---

- Regular speaker at UKOUG, SIOUG, DOAG, MOTS etc.
- My Blog: Oracle related stuff  
<http://oracle-randolf.blogspot.com>
- Maintainer of SQLTools++   
<http://www.sqltools-plusplus.org:7676/>  
<http://sqltpp.sourceforge.net>

# Who am I

Co-author of  
the latest  
OakTable book  
"Expert  
Oracle  
Practices"



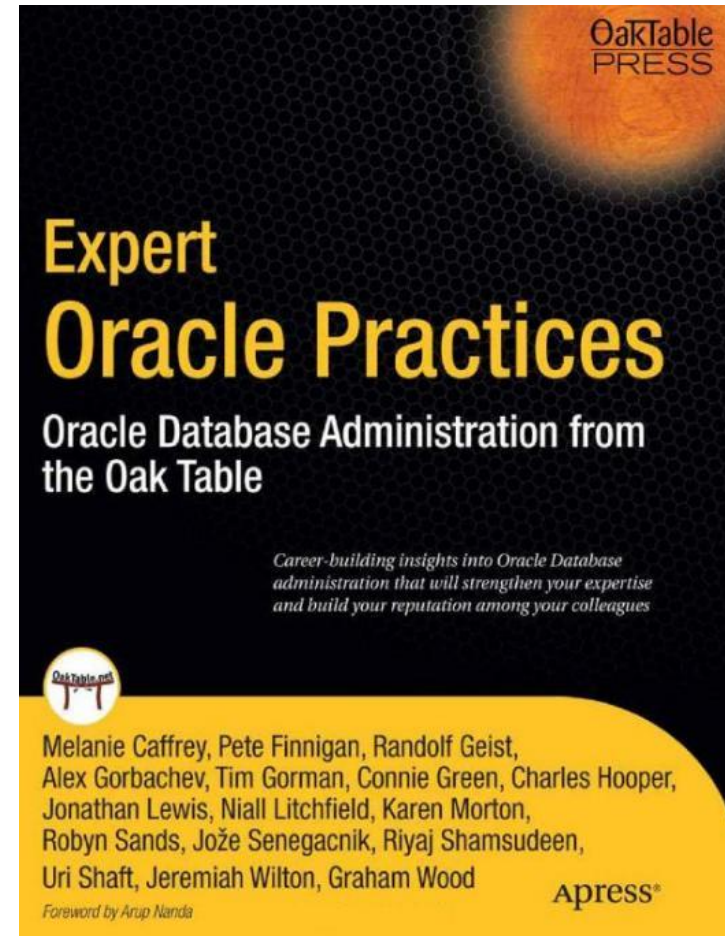
# Highlights

---

- Introduction to Troubleshooting vs. Tuning
- Basic Troubleshooting techniques - recap
- Active Session History - ASH
- Resource Contention: Latches / Mutexes
- Logical I/O investigations: Excessive consistent gets
- Tracing on O/S Level
- Oracle Heapdumps / Errorstacks / Systemstate dumps etc.
- Questions and Answers

# Advanced Oracle Troubleshooting

Loosely based  
on my chapter 8  
“Understanding  
Performance  
Optimization  
Methods”



# Advanced Oracle Troubleshooting

---

Extensive usage of Tanel  
Poder's tools

<http://tech.e2sn.com>

<http://blog.tanelpoder.com>

Download his toolset at:

<http://tech.e2sn.com/oracle-scripts-and-tools>

# Troubleshooting vs. Tuning

---

- Troubleshooting is not the same as tuning
- Troubleshooting is identifying the root cause of why something is taking longer than expected
- Tuning means fixing this identified root cause
- Finding a good tuning fix without any side effects is not a trivial task – think about “adding an index”
- This presentation focuses solely on the “Troubleshooting” part

# Troubleshooting Basics

---

- Performance is about time – response time, not about certain metrics
- Identify the issue by a clear problem statement
- Good: “This end-user report takes 10 times longer than expected”
- Not good: “The number of buffer gets is too high”

# Troubleshooting Basics

---

- Avoid looking at the system from an aggregated level (metrics, ratios)
- AWR / STATSPACK can be useful, if:
  - You plan to increase efficiency of the whole system (avoid hardware upgrades)
  - Your process is exclusively running on the system
- Systematic approach – start with the business process affected most

# Troubleshooting Basics

---

- Identify the affected process ideally by good application instrumentation or end user feedback !
- You can also try to identify via Top Activity / Sessions / SQL
- If you don't have an diagnostic license you can try MOATS  
(**M**)other (**O**)f (**A**)ll (**T**)uning (**S**)cripts  
<http://www.oracle-developer.net/utilities.php>

# Troubleshooting Basics

---

- Once process is identified, first things to check:
  - Session Statistics and Wait Interface => Best tool: Tanel Poder's "snapper"
  - Extremely powerful and flexible, no installation
- If you have identified SQLs or sessions that are taking longer than expected, use proven technology:
  - Extended SQL trace
  - DBMS\_XPLAN.DISPLAY\_CURSOR (>= 10g)
  - (Active Session History / ASH, more on that later)

# Troubleshooting Basics

---

- These two tools are complementary: Both tell you where most of the time is spent
- Extended SQL trace with waits enabled shows additionally the wait events, scope is session
- DBMS\_XPLAN.DISPLAY\_CURSOR shows the estimated cardinalities together with the actual ones (Starts + Rows), scope is a single SQL
  - Furthermore it shows the estimated memory requirements along with the actual memory used and the amount of TEMP space used when spilling to disk

# Troubleshooting Basics

---

- Bad execution plans are often caused by incorrect cardinality estimates
  - Check the estimated and actual cardinalities reported by `DBMS_XPLAN.DISPLAY_CURSOR`

# Troubleshooting Basics

---

- 11g added SQL Real-Time Monitoring
- Requires additional tuning & diagnostics pack license

- V\$SQL\_MONITOR

- V\$SQL\_PLAN\_MONITOR

- DBMS\_SQLTUNE.REPORT\_SQL\_MONITOR:

Combines information from  
V\$SQL\_PLAN\_MONITOR,  
V\$SESSION\_LONGOPS and  
V\$ACTIVE\_SESSION\_HISTORY

Loads of parameters and options - worth to study the details

# Pimp Up Your Basic Tools

---

- Adrian Billington's XPLAN wrapper for DBMS\_XPLAN.DISPLAY\* functions
- Use alternative trace file analyzers in addition to TKPROF:
  - Oracle's own Trace Analyzer (TRCA / TRCANLZR) See MOS Note 224270.1, and see also extended Explain Plan 215187.1 SQLT (SQLTXPLAIN)
  - Both tools require an installation and are a bit cumbersome to use, however they provide a wealth of detailed information

# Pimp Up Your Basic Tools

---

- More free trace file analyzers / tools:
  - Christian Antognini's TVD\$XTAT (<http://antognini.ch/top>)
  - Alberto Dell'Era's XPLAN / XTRACE tools (<http://www.adellera.it/>)
  - OraSRP (<http://www.oradba.ru/orasrp>)
  - SQLDeveloper (!)
- Support your developers by using Method R's MR Trace SQLDeveloper plug-in - allows to automatically fetch trace files from the server
- Or write your own little helper tool that transports trace files to the client automatically

# Advanced Oracle Troubleshooting

---

Demo Time!

basic\_troubleshooting\_summary.sql  
snapper\_to\_trace.sql

# Active Session History

---

- Oracle 10g has added Active Session History (ASH)
- No rocket science – simply takes a sample of all active sessions once a second
- Goldmine for troubleshooting, in particular for production issues that are transient / hard to reproduce

# Active Session History

---

- Interfaces:  
V\$ACTIVE\_SESSION\_HISTORY and  
DBA\_HIST\_ACTIVE\_SESS\_HISTORY
- Standard report available:  
RDBMS/ADMIN/ASHRPT.SQL
- Customized queries / reports possible
- Requires additional license:  
Diagnostic pack

# Active Session History

---

- The good news is that Active Session History is not rocket science, in principle it is about sampling V\$SESSION
- Therefore you don't need necessarily an additional license, but can do it yourself or use instead Kyle Hailey's S-ASH scripts for an "Active Session History" without ASH license  
<http://ashmasters.com>
- See also Oracle ASH project on Sourceforge by Marcin Przepiorowski  
<http://sourceforge.net/projects/orasash/>  
<http://oracleprof.blogspot.com/>

# Advanced Oracle Troubleshooting

---

Demo Time!

# Resource Contention: Latches / Mutexes

---

- If multiple processes compete for certain resources you might see so called "latch / mutex contention"
- Latches and Mutexes protect and serialize access to in-memory structures of the SGA
- So Latches and Mutexes are effectively "light-weight" locks

# Resource Contention: Latches / Mutexes

---

- Most commonly "latch cache buffer chain" contention when multiple processes excessively attempt to access buffers protected by the same latch buffer chain
- Another common issue is library cache contention due to excessive (hard) parsing activity

# Resource Contention: Latches / Mutexes

---

- Note that latch contention is usually a symptom rather than a cause
- Since spinning on latches burns CPU, excessive buffer gets can significantly increase CPU load
- On the other hand if you overload your CPUs then latch contention can be exaggerated by this CPU overload since the latches are held for too long and therefore contention of latches will be observed

# Resource Contention: Latches / Mutexes

---

- So latch contention can cause CPU load but can also be a symptom of CPU overload
- In general there is not much you can do about this except for changing the application logic
- In case of excessive buffer gets check if a bad statement execution plan causes this (mostly plans including NESTED LOOP operations)

# Resource Contention: Latches / Mutexes

---

- Details about latch/mutex activity can be obtained from dynamic performance views and internal X\$ fixed tables
- Tanel Poder's tool set comes handy: LATCHPROF / LATCHPROFX (only as SYS or when X\$ fixed views are accessible)

# Resource Contention: Latches / Mutexes

---

- In 10.2 and 11g library cache is protected by Mutexes rather than latches, therefore details can be obtained via  
`V$MUTEX_SLEEP_HISTORY`
- Use `MUTEXPROF` instead of `LATCHPROF/X` in that case
- More information: Tanel Poder's presentations and blog posts

# Advanced Oracle Troubleshooting

---

Demo Time!

`latch_contention.sql`

`library_cache_contention.sql`

# Logical I/O investigations: Excessive consistent gets

---

- Tracing and `DBMS_XPLAN.DISPLAY_CURSOR` tell you the number of consistent gets performed, but don't tell you the reason why they have been performed
- If you suspect that the number of consistent gets is higher than expected, it would be interesting to know the "reason" why the consistent get was performed

# Logical I/O investigations: Excessive consistent gets

---

- Excessive consistent gets can be caused by different reasons
- The most obvious reasons:
  - Bad execution plans
  - Reconstructing of older versions of a block by applying undo
  - Bugs, e.g. ASSM with non-default block sizes

# Logical I/O investigations: Excessive consistent gets

---

- Up to version 10g internal X\$ tables can be used to obtain information about the reason
- For 11g the internal structure has changed the contents are no longer updated

# Logical I/O investigations: Excessive consistent gets

---

- If you connect with a user capable of accessing X\$ tables, snapper can show you this as well
- Jonathan Lewis provides a nice script which takes a snapshot of this structure and outputs the delta afterwards  
[http://www.jlcomp.demon.co.uk/buffer\\_usage.html](http://www.jlcomp.demon.co.uk/buffer_usage.html)
- Alternatively use event 10200/10202 to trace consistent gets (11g)

# Advanced Oracle Troubleshooting

---

Demo Time!

`excessive_consistent_gets.sql`

# Leaving the instrumented area

---

- Now we'll come to the point where above tools do not allow a detailed diagnosis any longer
- You hit the limits for example if you have a case that:
  - consistently sits on CPU, but doesn't show anything in the session statistics, so you don't have a clue what the process is actually doing
  - performs some uninstrumented code section and therefore doesn't update the wait interface and/or the session statistics accordingly

# Leaving the instrumented area

---

- Examples:
  - PL/SQL processing
  - Parsing bugs
  - Instrumentation bugs, for example external table access up to 11.1.0.6
- In these cases a different approach is required if we need more details

# Leaving the instrumented area

---

- Trace the process on OS level using:
  - Tanel Poder's OSStackProf
  - DTrace (Solaris)
  - procstack (AIX)
  - gdb backtrace (Linux)
  - In future may be: probevue ( $\geq$  AIX 6.1), systemtap (Linux)

# Leaving the instrumented area

---

- Sometimes it is also useful to find out the system calls performed by the process using:
  - truss (Solaris, AIX)
  - strace (Linux)
  - ptrace
- Use My Oracle Support document **175982.1** to get an idea about the purpose of the functions traced

# Advanced Oracle Troubleshooting

---

Demo Time!

tpt\_public/aot/demo4.sql  
ostackprof.sql

# O/S Explain

---

- Tracing the process can also be used to create an Explain Plan on OS level to find out if a particular plan operation is responsible for most of the execution time (for example excessive nested loop operations)
- Tanel Poder's `os_explain` shell script

# Advanced Oracle Troubleshooting

---

Demo Time!

basic\_troubleshooting\_sample.sql  
os\_expl\_short\_stack.sql

# Oracle Heapdumps / Errorstacks / Systemstate etc.

---

- One particular helpful feature of the dumps is reading current bind values. Since these are stored in the process memory private to the process these cannot be accessed easily. With SQL Monitoring in 11g these are copied to shared memory structures, but we can help ourselves with certain DUMPs

# Advanced Oracle Troubleshooting

---

Demo Time!

basic\_troubleshooting\_sample.sql  
cursordump.sql

# Oracle Heapdumps / Errorstacks / Systemstate etc.

---

- If you can't tell from the call stack what is going on, sometimes it might be helpful to analyze the process using the available dumps, in particular if you see suspicious activity in the session statistics (for example, ever increasing pga / uga memory consumption)

# Advanced Oracle Troubleshooting

---

Demo Time!

xmlDom\_pga\_uga\_memory\_fragmentation\_demo.sql  
pga\_heap.sql

# Summary

---

- Define clear and concise problem statement
- Performance is about time
- Systematic approach
- Know your basic toolset
- Use advanced techniques only if instrumentation is not sufficient

# Advanced Oracle Troubleshooting

---

Questions

&

Answers

# Reference

---

- My Oracle Support
  - TRCANLZR: 224270.1
  - SQLTXPLAIN: 215187.1
  - Kernel functions: 175982.1
- Jonathan Lewis
  - [http://www.jlcomp.demon.co.uk/buffer\\_usage.html](http://www.jlcomp.demon.co.uk/buffer_usage.html)
- Tanel Poder
  - <http://tech.e2sn.com> & <http://blog.tanelpoder.com>
  - <http://tech.e2sn.com/oracle-scripts-and-tools>
- Adrian Billington
  - <http://www.oracle-developer.net>
- Oracle ASH / S-ASH
  - <http://sourceforge.net/projects/orasash/> & <http://oracleprof.blogspot.com/>
  - <http://www.ashmasters.com>

# Advanced Oracle Troubleshooting

---

Thank you!

<http://oracle-randolf.blogspot.com/>

<http://www.sqltools-plusplus.org:7676/>  
[info@sqltools-plusplus.org](mailto:info@sqltools-plusplus.org)