NoCOUG 2006

Advanced Research Techniques in Oracle

Tanel Põder
http://integrid.info

Introduction

- Name: Tanel Põder
- Occupation: Independent consultant
- Company: integrid.info
- Oracle experience: 8 years as DBA
- Oracle Certified Master
- OakTable Network Member
- EMEA Oracle User Group director

- This presentation is about less known and possibly unsupported *research/problem diagnosis* techniques
- Try this at home! (meaning *not* in your most critical production environment)

What is research?

- Dictionary.com:
  - Scholarly or scientific investigation or inquiry
  - Close, careful study
- www.cogsci.princeton.edu
  - Systematic investigation to estabilish facts
- www.mco.edu
  - Any systematic investigation, including research development, testing and evaluation, designed to develop or contribute to generalizable knowledge
- www.integrid.info
  - Finding out how things work ;)

Successful Research Prerequisites

- Interest
  - Work related (e.g. fixing performance issues or resolving corruptions/crashes)
  - Writing a paper/book
  - Pure interest on *how things work*
- Creativity
  - Although Oracle provides a lot of peep-holes into database internals, it pretty much remains a a black box for us
  - Thus creativity is needed combining various techniques
- Time
  - Tests and experiments should be as simple as possible, easily re-runnable and reproducible

Common Oracle Research Methods

- ~~Guesswork, ignorance~~
- Data Dictionary, DBA_, V$
  - VSQL, VSQLAREA
  - V$SYSSTAT, V$SESSTAT
  - V$SYSTEM_EVENT, V$WAITSTAT
  - V$SESSION_EVENT, V$SESSION_WAIT, V$SESSION_WAIT_HISTORY
- Statspack, Autotrace
- 10046 trace (sql_trace)
  - With binds and waits
- Various memory dumps, blockdumps
- 10g new stuff: ASH, AWR, ADDM

Shortcomings Of Current Methods

- NB! Current methods are sufficient for performance diagnosis in most cases
  - However there will always be special cases where “more” is needed
- V$ views show wide range db stats, but...
  - V$SESSION_EVENT stats are aggregated
  - V$SESSION_WAIT cannot be sampled too frequently (direct SGA attach would be needed)
  - V$SESSION_WAIT_HISTORY too “short”
- 10046 trace is great and pretty accurate...
  - But only SQL_TRACE, waits and binds
  - Performance/storage overhead in active environments

Improving File Based Tracing Framework

- Allow researcher to get immediate feedback from tracefiles
- Allow to process tracefiles on the fly
- Store only interesting parts of a tracefile
- Allow researcher to save time, by having easily re-runnable and comparable tests

```
SQL> select count(*) from t;
WAIT #5: nam='db file sequential read' ela= 89 p1=4 p2=195 p3=1
WAIT #6: nam='db file scattered read' ela= 213 p1=4 p2=196 p3=5
WAIT #6: nam='db file scattered read' ela= 10729 p1=4 p2=201 ...
WAIT #4: nam='SQL*Net message to client' ela= 3 p1=1650815232...

  COUNT(*)
----------
      1000
SQL>
```

tracedemo1.sql

Processing trace on the fly using pipes

- Unix only
  - Tried once with cygwin tail -f, didn't succeed

- Howto:
- Identify tracefile name
- Create *named pipe* in place of the file, *before* Oracle tries to open it
  - using mknod <name> p command
- Start process which reads from pipe
  - grep for example
  - watch out for output buffering
- Start tracing

Displaying processed trace output

```
SQL> select spid from v$process where addr = (
  2       select paddr from v$session where sid =
  3           (select sid from v$mystat where rownum = 1)
  4  );

SPID
------------
11191

SQL> host mknod /home/oracle/ORCL/udump/orcl_ora_11191.trc p
SQL> set define off
SQL> host grep "WAIT" /home/oracle/ORCL/udump/orcl_ora_11191.trc &
SQL> set define on
SQL> alter session set events '10046 trace name context forever,
   level 8';
Session altered.

SQL> select * from dual;
WAIT #12: nam='SQL*Net message from client' ela= 4805067
   p1=1650815232 p2=1 p3=0
...
```

Tracing logical IOs

- 10046 level 8 can trace physical IOs using wait interface
  - Physical IO in Oracle means system call operation to get a datablock from OS
  - This datablock might be in OS buffer cache or disk array cache, not exactly being “physical”
- Consistent gets can be traced using event 10200
  - Don't know any event for current gets though

```
SQL> @traceon 10200 1 "consistent read"
SQL> select * from dual;
Consistent read started for block 0 : 00400742
Consistent read finished for block 0 : 400742
Consistent read finished for block 0 : 400742
```

crdemo.sql

Tracing Enqueue Operations

- When resolving enqueue contention issues, it's often hard to find out who's *causing* high enqueue usage:
- 10046 trace shows only waits on enqueues
- X$KSQST
  - Kernel Service enQueue STatistics
  - Only system level aggregated information
- V$ENQUEUE_STAT[ISTICS]
  - Available from 9i, based on X$KSQST
  - Same problems as with X$KSQST
- Event 10704 traces local enqueue ops.
  - 10706 for Global enqueue operations in RAC
  - Also _ksi_trace could be used for GES tracing

`enqueuedemo.sql`

Information Sources

- Documentation, mailinglists, Google
- oraus.msg, $OH/rdbms/admin/*.sql
- Metalink / bug descriptions
- V$FIXED_VIEW_DEFINITION
  - Search by view name or definition
- V$TYPE_SIZE, V$LOCK_TYPE
- V$FIXED_TABLE, X$KQFCO
  - All X$ tables and their columns
- X$KSPPI
  - Parameters
- X$KSMFSV
  - Fixed SGA variables, pointers to various arrays
- oradebug help

Public documents and people

- It's worth to read the documentation first
  - There's much more information than I “had always thought”
- Google can give some surprising results
  - If search is specific enough
- Some websites:
  - http://www.ixora.com.au
  - http://www.jlcomp.demon.co.uk
  - http://www.juliandyke.com
  - http://www.hotsos.com
  - Oh yes, http://integrid.info too ;)
- Mailinglists & discussion boards
  - Oracle-L
  - comp.databases.oracle.server

$OH/rdbms/mesg/oraus.msg

- Descriptions for some events:

```
10046, 00000, "enable SQL statement timing"
// *Cause:
// *Action:

10053, 00000, "CBO Enable optimizer trace"
// *Cause:
// *Action:

10231, 00000, "skip corrupted blocks on
   _table_scans_"
// *Cause:
// *Action: such blocks are skipped in table
   scans, and listed in trace files
```

V$ View definitions

```
SQL> select table_name from dba_synonyms
  2  where synonym_name = 'V$INSTANCE';
TABLE_NAME
------------------------------
V_$INSTANCE

SQL> select text from dba_views where view_name = 'V_$INSTANCE';
TEXT
-------------------------------------------------------------------
select "INSTANCE_NUMBER","INSTANCE_NAME","HOST_NAME","VERSION",
  "STARTUP_TIME","STATUS","PARALLEL","THREAD#","ARCHIVER",
  "LOG_SWITCH_WAIT","LOGINS","SHUTDOWN_PENDING",
  "DATABASE_STATUS","INSTANCE_ROLE","ACTIVE_STATE"
from v$instance

SQL> select view_definition from v$fixed_view_definition where
   view_name = 'GV$INSTANCE';
VIEW_DEFINITION
-------------------------------------------------------------------
select ks.inst_id,ksuxsins,ksuxssid,ksuxshst,ksuxsver,ksuxstim,deco
de(ksuxssts,0,'STARTED',1,'MOUNTED',2,'OPEN',3,'OPEN MIGRATE','UNKN
...
e,0,'NORMAL',1,'QUIESCING',2,'QUIESCED','UNKNOWN') from x$ksuxsinst
 ks, x$kvit kv, x$quiesce qu where kvittag = 'kcbwst'
```

v.sql f.sql

Getting parameter information

- Interview question:
  - Query us all Oracle instance parameters and their values. Which view would you use?

  - V$PARAMETER
  - V$PARAMETER2
  - V$SYSTEM_PARAMETER
  - V$SYSTEM_PARAMETER2
  - show parameter
  - Join: X$KSPPI <-> X$KSPPSV

```
SQL> select count(*) from x$ksppi;
COUNT(*)
--------
    1173
```

Getting parameter information

- X$KSPPI - all Oracle instance parameters
- X$KSPPCV - current (session) values
- X$KSPPSV - system (instance) values
- X$KSPPCV2, X$KSPPSV2 - shows duplicate parameter values on separate lines
- Simple script (p.sql):

```
select n.ksppinm name, c.ksppstvl value,
       n.ksppdesc descr
from x$ksppi n, x$ksppcv c
where n.indx=c.indx
and n.ksppinm like '%&1%';
```

p.sql, pd.sql

oradebug help

```
SQL> oradebug help
HELP               [command]                 Describe one or all com
                                             mands
SETMYPID                                     Debug current process
SETOSPID           <ospid>                   Set OS pid of process to
                                             debug
SETORAPID          <orapid> ['force']        Set Oracle pid of process
                                             to debug
DUMP               <dump_name> <lvl> [addr]  Invoke named dump
DUMPSGA            [bytes]                   Dump fixed SGA
DUMPLIST                                     Print a list of available
                                             dumps
EVENT              <text>                    Set trace event in process
SESSION_EVENT      <text>                    Set trace event in session
DUMPVAR            <p|s|uga> <name> [level]  Print/dump a fixed
                                             PGA/SGA/UGA variable
SETVAR             <p|s|uga> <name> <value>  Modify a fixed PGA/SGA/UGA
                                             variable
PEEK               <addr> <len> [level]      Print/Dump memory
POKE               <addr> <len> <value>      Modify memory
WAKEUP             <orapid>                  Wake up Oracle process
SUSPEND                                      Suspend execution
RESUME                                       Resume execution
FLUSH                                        Flush pending writes to
                                             trace file
CLOSE_TRACE                                  Close trace file
TRACEFILE_NAME                               Get name of trace file
```

Suspending Oracle for analysis

- Suspending single process
  - oradebug suspend
  - kill -SIGTSTP, -SIGSTOP and -SIGCONT
  - tricks with alter session enable resumable, stopping archiver, suspending client, etc..
- Suspending an instance
  - kill -SIGTSTP command on all processes
  - flash freeze: oradebug ffbegin
  - end freeze: oradebug ffresumeinst
- Suspending whole RAC cluster
  - oradebug setinst all
  - oradebug ffbegin

Setting watchpoints

- Watchpoint helps to log any changes to given memory region:

-
  ```
  oradebug watch <addr> <length> \
                       <self|exist|all|target>
  ```

-
  ```
  oradebug show <local|global|target> wathcpoints
  ```

```
SQL> oradebug watch 0x50048B54 4 self
Local watchpoint 0 created on region [0x50048B54,
   0x50048B58).

ksdxwinit: initialize OSD requested
ksdxwcwpt: creating watchpoint on 0x0x50048b54, 4
           with mode 1
M:1110109332886167000:0x50048B54:4:0x088537DD:0x0
   8863F7A:0x0A486EB6:0x0A483948:ff000000
M:1110109332886972000:0x50048B54:4:0x0885555C:0x0
   88640A0:0x0A48761D:0x0A483948:00000000
M:1110109332890726000:0x50048B54:4:0x088537DD:0x0
   8863F7A:0x0A486EB6:0x08EC23CC:ff000000
```

Setting watchpoints

- Watchpoint helps to log any changes to given memory region:

```
                                  4-byte
memory?                        platform
^|--- timestamp ---  | modified |^| modifying |
 |                   |  address | | instr.addr/
M:1110109332886167000:0x50048B54:4:0x088537DD:
  0x08863F7A:0x0A486EB6:0x0A483948:ff000000
  caller     prev       prev       new value
             caller     caller
$ addr2line -e $OH/bin/oracle -f 0x088537DD
kslgetl
$ addr2line -e $OH/bin/oracle -f 0x08863F7A
ksfglt
$ addr2line -e $OH/bin/oracle -f 0x0A486EB6
kghalo
$ addr2line -e $OH/bin/oracle -f 0x0A483948
kghgex
```

- *_ksdxw_stack_depth* sets the dump stack depth

Calling kernel functions with oradebug

- oradebug call <function name> <params>
  - Does not work on all platforms (Windows, AIX)
  - On those, functions could be called using their starting memory address
- Finding function names
  - Errorstacks, gotten from Oracle or w. pstack/db
  - nm - a standard unix utility
  - objdump
  - oradebug SKDSTTPCS - translate proc calls

```
$ nm $ORACLE_HOME/bin/oracle > symbols.txt
$ grep dmp symbols.txt
oracle:08428c0a T aopdmp
oracle:094b8956 T curdmp
oracle:0aea2f82 t dreetdmp
...
```

Calling kernel functions with oradebug

- oradebug setorapid <xx>
- oradebug call curdmp
    - dumps open cursor information

```
SQL> oradebug setorapid 17
Unix process pid: 2763, image: oracle@local-
   host.localdomain (TNS V1-V3)
SQL> oradebug call curdmp
Function returned 0

****************  Cursor Dump  ******************
Current cursor: 2, pgadep: 0   pgactx: 526fbb24
   ctxcbk: 0 ctxqbc: 0 ctxrws: 52a01ae8
Cursor Dump:
----------------------------------------
Cursor 2 (b6b60288): CURROW  curiob: b6b6c340
 curflg: 46 curpar: 0 curusr: 0 curses 5421f924
 cursor name: select rownum, object_id from t
 child pin: 52eba3a0, child lock: 52eb5900,
   parent lock: 52ea4d68
```

Finding usable kernel functions

```
$ egrep -e "get|gt|dmp|dump" symbols.txt|grep ksm
SQL> oradebug setmypid
Statement processed.
SQL> oradebug call ksmget_sgamaxalloc
Function returned 3CBF94

SQL> oradebug call ksmgsizeof_granule
Function returned 400000
SQL> select to_number('400000', 'XXXXXX') from
   dual;
TO_NUMBER('400000','XXXXXX')
----------------------------
                     4194304
SQL> @pd ksm%granule
NAME                                     VALUE
------------------------------------ ------------
DESCR
-------------------------------------------------
_ksmg_granule_size                       4194304
granule size in bytes
```

Calling kernel functions with OS debugger

```
SQL> @i
USERNAME         SID SERIAL# SPID            OPID
------------ ------- ------- ------------ -------
SYS               41       8 2337              17

$ gdb

(gdb) attach 2337
Attaching to program: /
   home/oracle/product/10.1.0/bin/oracle, process
   2337
[New Thread -1229436992 (LWP 2337)]
Symbols already loaded for /
   home/oracle/product/10.1.0/lib/libunwind.so.3
...
(gdb) call kslgetl (0x50048B54,1)
$5 = 1
(gdb) call kslfre (0x50048B54)
$6 = 0
```

Invoking OS debugger on an event

- Could be used for invoking any executable/script:

```
$ cat /tmp/debug.sh
/bin/echo Hello World! $*
$ chmod u+x /tmp/debug.sh

SQL> alter system set
   "_oradbg_pathname"='/tmp/debug.sh';
System altered.
SQL> alter system set events 'logon debug forever';
$ sqlplus "/ as sysdba"
SQL*Plus: Release 10.1.0.3.0 - Production on Sun Mar 6
   09:41:59 2005
Copyright (c) 1982, 2004, Oracle.  All rights
   reserved.
Hello World! 28826
Connected to:
Oracle Database 10g Enterprise Edition Release
   10.1.0.3.0 - Production
With the Partitioning and Data Mining options
```

Invoking OS debugger on an event

- Getting process memory map on sort begin and end:

```
$ cat /tmp/debug.sh
/usr/bin/pmap $1 | /bin/grep mapped

SQL> set pause on
SQL> alter session set events '10032 debug forever';
Session altered.

SQL> select * from t order by 2;
mapped: 146928 KB  writable/private: 16328 KB
   shared: 73728 KB
mapped: 146928 KB  writable/private: 16328 KB
   shared: 73728 KB
SQL> alter session set events '10032 debug off';
```

- Could also use pstack, gdb and so on

Tracing system calls

- Really simple:
  - truss program, strace program

```
$ strace pwd
execve("/bin/pwd", ["pwd"], [/* 29 vars */]) = 0
uname({sys="Linux", node="localhost.localdomain",
   open("/etc/ld.so.preload", O_RDONLY)    = -1
   ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644,
   st_size=31589, ...}) = 0
old_mmap(NULL, 31589, PROT_READ, MAP_PRIVATE, 3, 0)
   = 0xb75e3000
close(3)                                = 0
open("/lib/tls/libc.so.6", O_RDONLY)    = 3
```

Attaching to running process:

- truss -p <spid> - Solarix, AIX
- strace -p <spid> - Linux
- trace -p <spid> - Tru64

Strace output

```
$ strace -p 2874
Process 2874 attached - interrupt to quit
read(7,
   "\0\267\0\0\6\0\0\0\0\0\21i\36\320\372\t\10\1\0\0\0\1
   \0"..., 2064) = 183
gettimeofday({1110107664, 205983}, NULL) = 0
getrusage(RUSAGE_SELF, {ru_utime={0, 80000}, ru_stime=
   {0, 190000}, ...}) = 0
...
statfs("/home/oracle/oradata/ORCL/system01.dbf",
   {f_type="EXT2_SUPER_MAGIC", f_bsize=4096,
   f_blocks=1008023, f_bfree=134407, f_bavail=83201,
   f_files=513024, f_ffree=420807, f_fsid={0, 0},
   f_namelen=255, f_frsize=0}) = 0
open("/home/oracle/oradata/ORCL/system01.dbf", O_RDWR|
   O_SYNC|O_LARGEFILE) = 12
gettimeofday({1110107664, 292448}, NULL) = 0
pread(12,
   "\6\242\0\0\243\300@\0\371\261\6\0\0\0\1\6a\10\0\0\2\
   0%"..., 8192, 403988480) = 8192
```

Transparent OCI call tracing

- Used-defined callback functions
  - Documented feature
  - Transparent, non-intrusive
  - Can do performance instrumentation
  - Pre-post processing, instead of processing
  - Works on Unix, Windows
  - Create dynamic load libraries
  - Register your dynamic callback functions
    - OCIUserCallbackRegister()
  - export ORA_OCI_UCBPKG=”lib1;lib2;lib3”
  - Run the application
- $ORACLE_HOME/rdbms/demo/ociucb.mk

  -
    ```
    make -f ociucb.mk user_callback \
    SHARED_LIBNAME=blah.so.1.0 OBJS=cdemoucbl.o
    ```

Peeking variables, dumping memory

- ORADEBUG PEEK
- ORADEBUG DUMP
- ORADEBUG DUMPVAR
- ORADEBUG DUMPTYPE
- X$KSMMEM

KST tracing and X$TRACE

- In-memory buffered tracing
- Only some events currently available
- Trace data is accessible either from:
  - X$TRACE
  - OS files dumped to disk
- Good for RAC and distributed issue tracing
- trace_enabled parameter must be true
- Syntax for enabling:
  - alter system set "_trace_events"='10000-10999:1:ALL';
  - Parameter 1: event list or range
  - Parameter 2: tracing level (0 - minimal, 255 verbose)
  - Parameter 3: process ID (<ID>, ALL, BGS, FGS)

KST tracing and X$TRACE

- Lots of parameters controlling KST tracing

```
trace_enabled                TRUE
_trace_processes             ALL
_trace_archive               FALSE
_trace_events                10000-10999:255:ALL
_trace_buffers               ALL:256
_trace_flush_processes       ALL
_trace_file_size             65536
_trace_options               text,multiple
_trace_buffer_wrap_timestamp TRUE

SQL> @xt
      TIME     seq#   event   op     sid
---------- -------- ------- ---- -------
1508545180   583506   10704   83      13
ksqgtl: acquire CU-2587046c-00000000 mode=X
   flags=SHORT why="contention"

1508545180   583507   10704   19      13
ksqgtl: SUCCESS
```

Conclusion

- It's amazing how many research and diagnosis features Oracle has!
  - How many more undiscovered features is there?
- On the other hand, do prefer conventional methods
  - No point in using hard-core internal mechanisms, if you can prove your point with simple methods
- Be careful when testing undocumented stuff and running Oracle code completely unconventionally
  - This stuff is only for experimental environments

Questions?

Thank you!

tanel@integrid.info

Tanel Põder
http://integrid.info
http://oaktable.net