

# Breaking Oracle



Simulating failures for testing  
and diagnostic practice

**Prepared By:**

---






**Jeremiah Wilton**  
**ORA-600 Consulting**

**Presented By:**

---

**Daniel Morgan**  
**University of Washington**

# About Dan Morgan

- Oracle Ace Director 
- University of Washington 
  - Wrote UW Oracle curricula
  - Primary program instructor - 9 years
- Author of Morgan's Library [www.psoug.org/library.html](http://www.psoug.org/library.html)
- Education Chair: PSOUG 
- Member: 
- Member:   
Oracle Applications Users Group
- Speaker: OOW, Collaborate, Kaleidoscope, EMEA, ...
- Working with Oracle since version 6

# About Jeremiah Wilton

- Amazon's first DBA 1997-2004
- Working with Oracle since 1994
- Owner, ORA-600 Consulting [www.ora-600.net](http://www.ora-600.net)
  - Architecture, scaling, performance
  - Availability, stability, complex recovery
  - Training, seminars, recruiting
- UW Certificate Program instructor
- Internals and nontrivial issue resolution



# Congratulation on 25 years

Countries					
Countries	Pages	Hits	Bandwidth		
United States	us	350563	628934	17.67 GB	
India	in	92840	170838	4.99 GB	
Great Britain	gb	50846	86890	2.50 GB	
Germany	de	45687	83207	2.57 GB	
France	fr	32790	59283	1.91 GB	
Canada	ca	31460	63047	1.53 GB	
Spain	es	26337	48181	1.40 GB	
Brazil	br	25733	53306	1.83 GB	
Australia	au	24531	46681	1.26 GB	
Italy	it	23534	44723	1.37 GB	
Poland	pl	16664	31917	968.22 MB	
Netherlands	nl	15370	30859	844.34 MB	
Switzerland	ch	14676	26832	799.18 MB	
Russian Federation	ru	12346	23506	987.28 MB	
Turkey	tr	9385	19470	807.62 MB	
Portugal	pt	9024	15923	397.86 MB	
China	cn	8986	20605	779.23 MB	
Mexico	mx	8515	17561	444.92 MB	
Belgium	be	7924	13649	384.56 MB	
Singapore	sg	7836	13228	355.30 MB	
Sweden	se	7674	13971	373.20 MB	
Romania	ro	7492	12698	333.29 MB	
Hong Kong	hk	7449	14047	394.17 MB	



and thank you for the warm welcome



# Oracle's Focus

- Data Center Failures
  - Data Guard, Streams, CDC
- Server Failure
  - RAC
- Storage Failure
  - ASM
  - Resumable Transactions
- Human Error
  - Flashback & Transaction Back-out
- And when all nothing else will suffice
  - RMAN

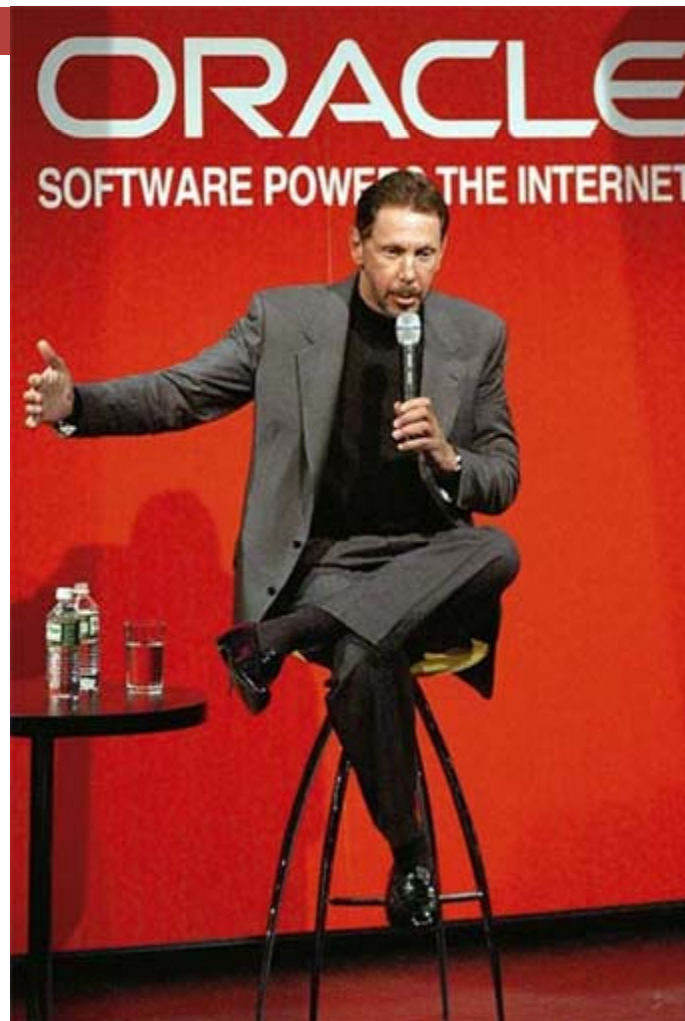
# Problem Profiles

- Hangs
  - Single-session
  - Multi-session
  - Whole instance
  - Multi-instance
- Spins
  - Server process
  - Background process
- Crashes
  - Session/server/process
  - Whole instance
  - ORA-600, ORA-7445
- Corruption/data loss
  - Files
  - Blocks
  - Logical
  - Diabolical

# Rationale

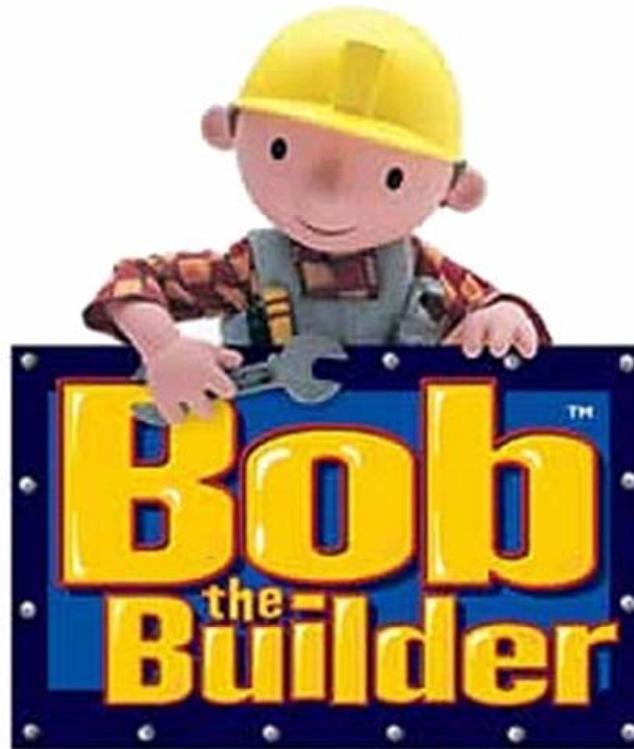
- Substitute for real-world ordeals
- Hard to find good troubleshooters
- High cost of outages
- Opportunity for improvement
- Obscurity of diagnostic skills
  - Not a standard DBA skill
  - Not well documented
- Inadequacy of Oracle Support Services first-line
- Fun, exciting

# Can We Break It?



# Yes We Can!

---



# Inducing Load

- Need a realistic load to induce hangs, etc.
- Resource contention is a problem of concurrency
- Under load, problems get worse
- Helps find scaling limits of a system
- An inactive site is no excuse for not learning
- Many recent options available

# Induced Load: Options

- Generated workload
  - Can be turned up to exhaust server resources
- Recorded workload
  - Your application's true load
  - Less opportunity to ratchet up
- Application service loaders
  - HP LoadRunner, OpenSTA
- Database-only loaders
  - Database Replay, HammerOra, Swingbench

# Swingbench

- Open-source tool by Dominic Giles (Oracle UK)
- Synthetic load harness
- Useful canned workloads
  - Order Entry
  - Calling Circle
- Possible to roll your own workload
- Quick and easy to set up
- <http://www.dominicgiles.com/swingbench>

# Database Replay

- Part of 11g Real Application Testing
- Capture from earlier versions
  - 9.2.0.8, 10.2.0.3, 10.2.0.4
- Allows workload to resemble real application
- Allows subsetting by user, app, etc.
- Premium option
- Primarily for change assurance

# Hangs

- One or more sessions getting "stuck"
- Really means waiting on something
  - A single hanging session can hold resources required by others so it can cascade
- Locks, latches, I/O, object serialization
- Hanging sessions may be holding resources needed by others
- Work ethic of waits: Don't ignore waits
- Long (legitimate) waits vs. hangs
  - Oracle's view
  - Customer's view

# Hangs

- Oracle's View

- Defined by Oracle's new 11g's Hang Detection
- Requires a chain of internal resources recognized by their tool and extending beyond three waiters

```
SELECT COUNT(*) FROM v$wait_chains;
```

- Customer's View

- Not as stringent
- Can you work? If not it is a hang!

# Hangs

```
SQL> SELECT state, wait time, seconds in wait  
2 FROM v$session_wait;
```

STATE	WAIT_TIME	SECONDS_IN_WAIT
WAITED SHORT TIME	-1	0
WAITING	0	2807
WAITING	0	3
WAITING	0	10
WAITING	0	2
WAITING	0	175
WAITING	0	10
WAITING	0	1
WAITING	0	1007
WAITING	0	167

# Hangs

- If more than 1 second they are not real  
wait\_time 0, 1, 2, is a decoded value related to state
- If wait\_time is -1 seconds\_in\_wait must be 0 or its lying
- One or more sessions in non-idle or non-instrumented waits

# Whole – Instance Hang

- Hang I/O calls by processes that can't time out

```
root@dbhost# mount -F nfs -o rw \  
localhost:/opt/oracle/oradata/od08/bct /mnt/orabct
```

```
SYS> alter database enable block change tracking  
using file '/mnt/orabct/bct.ora';
```

```
user@dbclient$ ./charbench
```

```
root@dbhost# /etc/init.d nfs.server stop
```

```
SYS> column program format a15 trunc  
SYS> column event format a45  
SYS> select sid, program, event, state,  
seconds_in_wait, blocking_session  
from v$session where type != 'BACKGROUND';
```

- CTWR holds resources needed by running sessions

# Spins

- Endless loops
- Any process consuming 99+% of CPU and not doing useful work
- Process may be hanging or not
- Found with top or ps
- If hanging may be holding resources needed by others

# Spins

- One place to look:

```
SELECT sid, event, state,  
seconds_in_wait  
from v$session  
where type <> 'BACKGROUND'  
ORDER BY by 4;
```

# Server Process Spins

- Hang and spin in regular expression search

```
SQL> select 1 from dual where regexp_like(' ','^*[ ]*a');
```

```
oracle@dbhost$ ps -eo pid,pcpu,args | sort -n +1 | tail -10
```

```
SQL> @waits
```

# Background Process Spins

- Spinning background procs can't always be killed without terminating the instance

```
oracle@db02$ ps -eo pid,s,args | grep ora_arc
oracle@db02$ kill -STOP `ps -eo pid,args | grep ora_arc \
    | grep -v grep | awk '{print $1}'`
oracle@db02$ ps -eo pid,s,args | grep ora_arc
SQL> select group#, sequence#, archived, status from v$log
    order by sequence#;
SQL> alter system switch logfile;
SQL> alter system switch logfile;
SQL> alter system switch logfile;
oracle@db02 $ ps -eo pid,pcpu,args | sort -n +1 | tail -10
SQL> column event format a45
SQL> select event, state, seconds_in_wait from v$session
    where type = 'BACKGROUND' and program like '%LGWR%';
```

# Crashes

- Server processes that exit abnormally
- For example:
  - ORA-03113
  - ORA-00600
    - Not always emergencies - might be a harmless bug
    - Don't panic / look in metalink / perhaps open an SR /
    - But does not necessarily equal corruption or failure
  - ORA-07445
    - Almost always causes a crash unless it is a background process

# Crashes

- Usually ORA-00600 and ORA-07445
- Single process crash *can* take down whole instance
- ORA-00600: internal error code, arguments: [] [] [] []
  - First argument tells you calling function or numeric identifier
  - Additional arguments provide more information
  - Process/session does not always die
  - Not necessarily an emergency
- ORA-07445: exception encountered: core dump [] []
  - Core dump
  - First argument tells you where in the code (10g+)
  - Second argument is the signal (kill -l)
  - Additional arguments provide more information

# 11g Background Processes

## Which ones crash the instance?

Process Name	Description
<b>ACMS</b>	<b>Atomic controlfile to memory server</b>
<b>ARCn</b>	Redo log archivers
<b>CJQn</b>	Job scheduler coordinator
<b>CKPT</b>	<b>Checkpoint</b>
<b>Dnnn</b>	Dispatchers
<b>DBRM</b>	<b>Resource manager process</b>
<b>DBWn</b>	<b>Database writer processes</b>
<b>DIA0</b>	Diagnosibility process 0
<b>DIAG</b>	Diagnosibility coordinator
<b>FDBA</b>	Flashback data archiver process
<b>Jnnn</b>	Job scheduler processes
<b>LGWR</b>	<b>Redo log writer</b>
<b>LMDn</b>	<b>Global enqueue service daemons</b>
<b>LMON</b>	<b>Global enqueue service monitor</b>
<b>MMAN</b>	<b>Memory manager</b>

Process Name	Description
<b>MMNL</b>	<b>Manageability Monitor Process 2</b>
<b>MMON</b>	<b>Manageability Monitor Process</b>
<b>PING</b>	<b>Interconnect latency measurement</b>
<b>PMON</b>	<b>Process monitor</b>
<b>PSPn</b>	<b>Process spawners</b>
<b>Qnnn</b>	<b>Queue cleanup processes</b>
<b>QMNC</b>	<b>Queue coordinator</b>
<b>RECO</b>	<b>Distributed recovery process</b>
<b>RMSn</b>	<b>RAC management server</b>
<b>RVWR</b>	<b>Recovery writer</b>
<b>Snnn</b>	<b>Shared servers</b>
<b>SMCO</b>	<b>Space management coordinator</b>
<b>SMON</b>	<b>System monitor process</b>
<b>VKTM</b>	<b>Virtual keeper of time process</b>
<b>Wnnn</b>	<b>Space management processes</b>

# ORA-00600 Example

- Simplest case in PL/SQL

```
SQL> declare
      a exception;
      pragma exception_init(a,-600);
begin
      raise a;
end;
```

- Nicer, lets you specify the arguments

```
SQL> oradebug unit_test dbke_test dde_flow_kge_ora ouch! 0 0
```

# Bug That Raises ORA-00600

- Bug 6073325: SELECT QUERY WITH CONNECT BY PRIOR FAILS WITH ORA-00600 [KKQCBYDRV:1]

```
SQL> select 1
      from sys.table_privileges tp, user_objects uo
      where tp.grantee in
            (select 1 from sys.dba_role_privs
             connect by prior granted_role = grantee
             start with grantee = 'scott');
```

- Raises ORA-600, but we are still connected
- Not all -600 errors are fatal (most are not)
- Just a unhandled exception - no reason to panic

# ORA-07445 Example

- Simplest case: send a signal

```
SQL> select spid from v$process p, v$session s
       where p.addr = paddr
       and sid = sys_context('USERENV','SID');
```

```
oracle@db02$ kill -SEGV 2513
```

- Use PL/SQL

```
SQL> declare
       a exception;
       pragma exception_init(a,-7445);
begin
       raise a;
end;
```

# Real ORA-07445 Bug

- Bug 6244173: ORA-07445 IN QEESTRAVERSEEXPR FOR HIERARCHICAL QUERY

```
SQL> create table t2(coll varchar2(60));
SQL> create table t1(c1 varchar2(60),
                    c2 varchar2(1),
                    c3 varchar2(60),
                    c4 varchar2(60));

SQL> explain plan for
      select 1 from t1 a, t2 b ,t1 c
      where b.coll = 'xxslc_department'
      and a.c1 not between c.c3 and c.c4
      start with a.c2='p'
      connect by prior a.c1 between a.c3 and a.c4;
```

- Raises ORA-3113, so we look in alert log...
- Nature of a crashed process to generate a disconnect
- Continued use of dead connection gives app:
  - ORA-3114: Not connected to Oracle
  - ORA-1041: internal error. hostdef extension doesn't exist
    - oerr ora 1041 - Call support!

# Whole - Instance Crashes

- Something causes a required background process to exit
- ORA-600, ORA-7445, I/O errors, etc.
  - Can actually be any error that prevents the next step
- Some restart and some crash the instance

# Instance Crashes

- Simple case: kill an essential background process (tail the alert log)

```
oracle@db02$ ps -eo pid,args | grep ora_ckpt | grep -v grep
oracle@db02$ kill -KILL <pid>
```

- Simple case: send a SIGSEGV or SIGBUS to an essential background process

```
oracle@db02$ ps -eo pid,args | grep ora_dbrm | grep -v grep
oracle@db02$ kill -SEGV <pid>
```

– Raises ORA-07445

# Instance Crashes

- Cause fatal errors in essential background processes

```
SQL> select pid, program, background  
       from v$process
```

```
       where background = 1;
```

```
SQL> oradebug setorapid 16
```

```
SQL> oradebug call kgeasnmierr 4455547624 18446744071472029760  
18446744071562043788 2 1 1
```

# Corruption

- Physical
  - File headers
  - Data blocks
  - Control files, log files, other logs
  - Caused by Oracle, O/S and hardware bugs
- Logical
  - Application tables
  - Data dictionary

# Data Block Corruption

- Simple example: garbage into a block
- Find a block in a known table

```
SQL> select min(dbms_rowid.rowid_block_number(rowid))
       from soe.customers;
```

```
SQL> select customer_id, cust_email from soe.customers
       where dbms_rowid.rowid_block_number(rowid) = 12;
```

```
oracle@db02$ dd if=/opt/oracle/oradata/od08/soe.dbf bs=8192 \
iseek=12 count=1 | strings | grep Sachin.Neeson@oracle.com
```

```
oracle@db02$ dd if=$ORACLE_HOME/bin/oracle \
of=/opt/oracle/oradata/od08/soe.dbf \
bs=8192 oseek=12 count=1 conv=notrunc
```

```
1+0 records in
```

```
1+0 records out
```

```
SQL> alter system checkpoint;
```

# Data Block Corruption

- Check the alert log - no errors!

- Read the block

```
SQL> select customer_id, cust_email from soe.customers
       where dbms_rowid.rowid_block_number(rowid) = 12;
```

```
SQL> alter system flush buffer_cache;
```

```
SQL> select customer_id, cust_email from soe.customers
       where dbms_rowid.rowid_block_number(rowid) = 12;
```

- Restore data block (read again)

```
RMAN> blockrecover datafile
       '/opt/oracle/oradata/od08/od08/soe.dbf' block 12;
```

# Other Vulnerable Files

---

- Archived redo logs
- Flashback logs
- Block Change Tracking file
- Backups

# Logical Corruption

- Erroneously changed data
  - Missing/incorrect predicate (where clause)
- Human error/application bug
- Oracle bug (wrong results)
  
- Many tools to resolve
  - Flashback query
  - Flashback transaction
  - Flashback table
  - Flashback database
  - Log Miner
  - Traditional point-in-time recovery
  - Mini-clone recovery (PITR of a partial DB)

# Logical Corruption

- User oops: missing where clause

```
SQL> update customers set cust_first_name = 'Nimrod'  
      where rownum < 1000;
```

```
SQL> commit;
```

```
SQL> select versions_startscn, versions_endscn, versions_xid  
      from customers  
      versions between timestamp sysdate-(.25/24) and sysdate  
      where cust_first_name = 'Nimrod';
```

```
SQL> select undo_sql from flashback_transaction_query  
      where xid = '00090015000003A1'
```

- Quality resolution requires examining "versions between" to get exact SCN of changes (undo\_retention).
- Don't forget that there may have been subsequent changes to rows

# Q&A

- Contact:

Dan Morgan

[damorgan@u.washington.edu](mailto:damorgan@u.washington.edu)

Morgan's Library

<http://www.psoug.org/library.html>

Jeremiah Wilton

[jeremiah@ora-600.net](mailto:jeremiah@ora-600.net)

<http://www.ora-600.net>

<http://oradeblog.blogspot.com>