NYOUG

# RDBMS Forensics

## Troubleshooting Using ASH

Tim Gorman

www.Delphix.com

# Agenda

- **Outright errors and failures leave evidence…**
  - In log files, in trace files, in core dump files
- Perfor [illegible]
  - No [illegible]
  - Trac [illegible]
- Alway [illegible] can provid [illegible]
  - I'd lik [illegible]
- Troubles [illegible] line
  - Understa [illegible] accumulate
  - Resisting th [illegible] others
  - Instead workin [illegible] the problem

# Agenda

- **Outright errors and failures leave evidence…**
  - In log files, in trace files, in core dump files
- **Performance problems are not errors**
  - No evidence left behind in log files
  - Trace files help only if you had the foresight to enable them
- Alway [illegible] can provid [illegible]
  - I'd lik [illegible] …
- Troubles [illegible] line
  - Understa [illegible] accumulate
  - Resisting th [illegible] others
  - Instead workin [illegible] the problem

# Agenda

- **Outright errors and failures leave evidence…**
  - In log files, in trace files, in core dump files
- **Performance problems are not errors**
  - No evidence left behind in log files
  - Trace files help only if you had the foresight to enable them
- **Always-on diagnostic features in the database can provide the needed evidence**
  - I’d like to offer two actual recent case studies as illustration…
- Troubles [illegible] line
  - Understa [illegible] at accumulate
  - Resisting th [illegible] others
  - Instead workin [illegible] the problem

# Agenda

- Outright errors and failures leave evidence…
  - In log files, in trace files, in core dump files
- Performance problems are not errors
  - No evidence left behind in log files
  - Trace files help only if you had the foresight to enable them
- Always-on diagnostic features in the database can provide the needed evidence
  - I'd like to offer two actual recent case studies as illustration…
- Troubleshooting requires patience and discipline
  - Understanding where evidence of from a problem might accumulate
  - Resisting the impulse to Google for the opinions of others
  - Instead working from the facts scattered around the problem

# First case study…

- **Informatica jobs reporting Oracle error (something about “*connection lost contact*”)**
  - Obtaining the error messages from the Informatica logs showed ORA-03135
  - Searching for that error message in MyOracleSupport (MOS) yields…
    - Note #730066.1 (entitled “***Diagnosis of ORA-3135/ORA-3136 Connection Timeouts when the Fault is in the Database***”)

# Researching in MOS…

- **Note #730066.1 indicates that ORA-03135 ("*connection lost contact*") is returned to the client-side of the database connection (i.e. Informatica PowerMart, in this case)**
    - **It also mentions that on the database side, error ORA-03136 *("inbound connection timed out")* is recorded to the database *alert.log* file as well…**

# Researching in MOS…

- **Note #730066.1 goes on to explain the sequence of SQL commands executed when establishing a database connection**
  - **Each of these SQL statements has to be parsed, executed, and fetched**
    - **Thus presenting the possibility of a “hang” if there is any kind of contention**

- **Also, [illegible] sympto[illegible] [illegible]rors**
  - **Almost [illegible]**

# Researching in MOS…

- Note #730066.1 goes on to explain the sequence of SQL commands executed when establishing a database connection
  - Each of these SQL statements has to be parsed, executed, and fetched
    - Thus presenting the possibility of a “hang” if there is any kind of contention
- Also, mentions two Shared Pool bugs with symptoms including ORA-03135/3136 errors
  - Almost goes without saying… ☺
  - Far too early to leap to the conclusion of a code bug…

# Researching in MOS…

- **Note #730066.1 then goes on to suggest three different methods for diagnosing "hangs" while establishing database connections…**

1. From a separate SYSDBA session in SQL*Plus, take three SYSTEMSTATE dumps each 90 seconds apart while the problems are happening
2. Examine *Active Session History* (ASH) information for the 15 minutes leading up to the occurrence of the error
3. Continuous monitoring of the contents of V$LATCHHOLDER while the problems are happening

# Researching in MOS…

- Also, note #465043.1 (entitled *"Troubleshooting ORA-3136 WARNING Inbound connection timed out")* is useful…
  - Further describing the error as a timeout specified by the SQL*Net parameter SQLNET.INBOUND_CONNECT_TIMEOUT (defaults to 60 secs)
    - Introduced in Oracle10g to prevent *Denial Of Service* (DoS) attacks on the database

# Researching on Google…

- **Searching for keywords “`oracle ora-3135 ora-3136`” mostly yields articles that repeat the following advice posted to an Oracle TechNet (OTN) discussion forum…**

```
It's a warning. by default there would be some time in
   seconds, usually a connection from client will get
   connected to DB Server with in prescribed time limit.
   If the connection does not happen then you can find
   the warning messages. In order to handle this you can
   include some parameters in sqlnet.ora and
   listener.ora

   Example :-

   sqlnet.ora
   SQLNET.INBOUND_CONNECT_TIMEOUT = 120

   In the listener.ora
   INBOUND_CONNECT_TIMEOUT_LISTENER = 110
```

# Research summary

- **Connections taking longer to establish than the timeout specified by SQL*Net timeout parameters**
  - On both instances of a 2-node RAC cluster, we have already set all these SQL*Net parameters to 300 secs (5 mins)
    - *Would it be reasonable to set these higher?*
  - Consider:
    - timeouts tend to turn a failure condition into a performance problem
    - longer timeouts show as bigger performance problems…

# Research summary

- **ORA-03136 errors only showed up in the *alert.log* file of RAC02**
  - **No error messages found in RAC01**

```
...
Mon Oct  4 22:05:44 2010
SUCCESS: diskgroup D3098_ORAARCH was dismounted
Mon Oct  4 22:07:05 2010
WARNING: inbound connection timed out (ORA-3136)
Mon Oct  4 22:07:05 2010
WARNING: inbound connection timed out (ORA-3136)
...
```

# Active Session History (ASH)

ASH is a slightly different take from other information in the *Automatic Workload Repository* (AWR)

- All AWR information is based on real-time statistics maintained in memory-based V$ performance views
  - V$ performance views come in two basic flavors…
    - Cumulative instance-level statistics
    - Cumulative session-level statistics ***for currently-active sessions only***
  - AWR is the long-term repository for the information in the V$ performance views
    - Only for the *cumulative instance-level statistics*…
    - **…*session-level statistics* are not retained past the end of the session…**

# Active Session History (ASH)

- Why is session-level data valuable?
  - Instance-level data is aggregated almost to the point of being meaningless for troubleshooting purposes
    - AWR data aggregated to an hour, instance-wide, by default
      - Good for finding "Top N Worst" SQL and general **trends** lasting more than an hour
      - Bad for finding **anomalies**, incidents happening in seconds or minutes
  - **Session-level data** shows more detail for anomalies
    - Includes information about the identity of the user or program, giving some idea of the context of a problem
  - Session-level data is similar to tracing or logs stored inside the database

# Active Session History (ASH)

- Circular ASH buffers are intended to store "*about an hour*" of ASH information
  - Oracle background MMNL process scans V$SESSION info in shared memory every second for "active" session information, flushes to ASH buffers in shared memory
    - Parameter "`_ash_sampling_interval`", defaults to 1000 millisecs (1 sec)
- Flushing from shared memory buffers to permanent AWR tables occurs…
  - At each AWR snapshot
  - When ASH buffer becomes 66% full
    - Parameter "`_ash_eflush_trigger`", defaults to "`66`" (percent)
- Not all rows in shared memory are flushed to permanent AWR tables
  - Parameter "`_ash_disk_filter_ratio`", defaults to "`10`" (percent)

# Active Session History (ASH)

- How long ASH data is *actually* retained in fixed size SGA buffers is dependent on database workload
  - More active sessions per second, less data can be retained
  - Fewer active sessions per second, more data can be retained longer
- ASH data retained in permanent AWR tables is configurable
  - Viewable in DBA_HIST_WR_CONTROL view
  - Using MODIFY_SNAPSHOT_SETTINGS procedure in DBMS_WORKLOAD_REPOSITORY package

# V$ACTIVE_SESSION_HISTORY

- **Columns include…**
  - Timestamp when row was collected
  - User ID, session/serial# ID, type, and status
  - Program, module, action (set using DBMS_APPLICATION_INFO)
  - SQL ID, SQL plan hash value, SQL opcode
  - Execution plan line ID, execution plan operation and options
  - PL/SQL module info (entry point, current procedure)
  - Parallel execution query coordinator (QC) information
  - Blocking session information, if waiting on enqueue
  - Wait event information with parameters

# DBA_HIST_ACTIVE_SESS_HISTORY

- **Contains same columns as V$A_S_H, plus…**
  - **AWR snapshot ID**
  - **DBID and instance number**
- **Data less dense than in V$A_S_H**
  - **every 10 seconds rather than every second**
- **Data retained longer than in V$A_S_H**
  - **Good for forensic investigation when working beyond the *horizon* of the data in V$A_S_H**

# So what does ASH show?

```
SQL> select  sample_time, program, event, p1, p2
  2  from    dba_hist_active_sess_history
  3  where   sample_time between '04-OCT-2010 22:06:30'
  4          and '04-OCT-2010 22:07:10'
  5  and     instance_number = 2
  6  order by sample_time;

SAMPLE_TM       PROGRAM              EVENT                     P1        P2
--------------- -------------------- ------------------------- --------- ------
04-OCT 22:06:30                      gc current request                1    180
04-OCT 22:06:30 RunDbJob.pl@corexprd row cache lock                   13      0
04-OCT 22:06:30 oracle@rac02 (P018)  PX Deq Credit: send blkd  268828671    128
04-OCT 22:06:30 oracle@rac02 (P008)  direct path read               3989 315200
04-OCT 22:06:30 oracle@rac02 (P016)  PX Deq Credit: send blkd  268828671    128
04-OCT 22:06:30 oracle@rac02 (P014)  direct path read               2971 136640
04-OCT 22:06:30 pmdtm@infaxprd (TNS  gc buffer busy                 3752 265709
04-OCT 22:06:30 oracle@rac02 (P022)  PX Deq Credit: send blkd  268828671    128
04-OCT 22:06:30                      enq: SQ - contention     1397817350    144
04-OCT 22:06:30 oracle@rac02 (P026)  PX Deq Credit: send blkd  268828671    128
04-OCT 22:06:30 oracle@rac02 (P012)  direct path read               2972 294976
04-OCT 22:06:30 oracle@rac02 (P034)  PX Deq Credit: send blkd  268828671    128
```

# Lots of concurrency waits

- Querying DBA_HIST_A_S_H to 30 seconds prior to logged ORA-03136 errors, then filtering out the "normal" I/O waits and PX-related waits, we have…
  - `enq: SQ – contention`
  - `row cache lock`
  - `gc current block 2-way`
  - `gc buffer busy`
  - `gc current request`
  - `latch: row cache objects`
- Remember that the long-term retained ASH information in the DBA_HIST_A_S_H view consists of 10-second samples
  - So we may have missed a lot possible culprits!

# enq: SQ – contention

- **This event means that a sequence object is “refreshing” its next set of sequence values from the data dictionary table SYS.SEQ$**

```
SAMPLE_TM       EVENT                 P1          P2
--------------- --------------------- ----------- ---
04-OCT 22:06:30 enq: SQ - contention  1397817350  144
```

- **According to the view V$EVENT_NAME, parameter P2 on this event contains an OBJECT_ID value**

# enq: SQ – contention

- **Translating OBJECT_ID = 144 using the DBA_OBJECTS view in the database shows…**

```
SQL> select owner, object_name, object_type
  2  from dba_objects where object_id = 144;

OWNER  OBJECT_NAME     OBJECT_TYPE
------ --------------- -------------------
SYS    AUDSES$         SEQUENCE
```

- **This sequence (AUDSES$) is used to generate values for the column AUDSES in the V$SESSION view when a database session is established…**

# row cache lock

- **This event indicates that modifying data dictionary information cached in the Shared Pool of the SGA (a.k.a. “row cache”) is encountering contention**

```
SAMPLE_TM        EVENT                   P1          P2
---------------- ----------------------- ----------- ---
04-OCT 22:06:30 row cache lock          13          0
```

- **According to the view V$EVENT_NAME, parameter P1 on this event contains a *cache ID* value**

# row cache lock

- **Querying the view V$ROWCACHE…**

```
SQL> select parameter from v$rowcache
  2  where cache# = 13;

PARAMETER
--------------------------
dc_sequences
```

**Hmmm… are we seeing a pattern?**

**..but wait – there's more!!!**

# gc_current_request

- **This event is posted when an instance must obtain the most current copy of a block buffer from another RAC instance…**

```
SAMPLE_TM       EVENT                      P1          P2
--------------- ---------------------- ----------- ---
04-OCT 22:06:30 gc current request     1           180
```

- **According to the view V$EVENT_NAME, parameter P1 is *file#* and P2 is *block#***

# gc_current_request

- **Querying the DBA_EXTENTS view, we can find the database object in which this contended-for block resides…**

```
SQL> select owner, segment_name from dba_extents
  2  where file_id = 1 and
  3  180 between block_id and (block_id + (blocks-
  1));

OWNER           SEGMENT_NAME
--------------- ---------------------
SYS             SEQ$
```

**Couldn’t we conclude that we have a problem with sequences?**

# Poorly-cached sequence?

- **Let's see if AUDSES$ needs a higher CACHE_SIZE…**

```
SQL> select sequence_owner,sequence_name,cache_size,last_number
  2  from dba_sequences
  3  where cache_size < 20
  4  and (nvl(min_value,0)+last_number)/increment_by  >= 10000
  5  order by (nvl(min_value,0)+last_number)/increment_by desc;

OWNER    SEQUENCE_NAME                  CACHE_SIZE LAST_NUMBER
-------- ------------------------------ ---------- -----------
SYS      DBMS_LOCK_ID                           10  1073741891
SYS      IDGEN1$                                10    18987301
SYS      WRI$_ADV_SEQ_MSGGROUP                  10     6428319
PROD     JL_SEQ                                  0     2344665
SYS      ORA_TQ_BASE$                            0     1895033
SYS      WRI$_ALERT_SEQUENCE                    10     1180544
MDSYS    TMP_COORD_OPS                           0     1000000
PROD     DL_SEQ                                  0      964612
PROD     RUN_ID_SEQ                              0      187744
SYS      SCHEDULER$_INSTANCE_S                  10      166467
SYS      WRI$_ADV_SEQ_TASK                      10      151284
SYS      OBJECT_GRANT                           10      139382
PROD     DF_SEQ                                  0       53402
```

# Poorly-cached sequence?

- **The sequence SYS.AUDSES$ is not in the list!**
- **In fact, the CACHE_SIZE attribute on the AUDSES$ sequence has already been increased…**

```
SQL> select cache_size from dba_sequences
  2  where sequence_name = ‘AUDSES$’;

CACHE_SIZE
----------
       500
```

- **It turns out that I had performed the exact same investigation 3 months previously**
  - **…and that I had jumped to the same conclusion!**

# Poorly-cached sequence?

- So, as much as it appears that cache refreshes by the AUDSES$ sequence are the cause of the ORA-03135/3136 errors
  - It is equally apparent that increasing CACHE_SIZE further does not resolve the problem

**What else could it be?**

# RAC contention on a block?

- **Does the AUDSES$ sequence reside within block #180 on file #1?**

```
SQL> declare
  2    v_rid   rowid; v_type number; v_obj number;
  3    v_rfno  number; v_rno number;
  4    v_bno   number;
  5  begin
  6    select rowid into v_rid from sys.seq$ where obj# = 144;
  7    dbms_rowid.rowid_info(v_rid, v_type, v_obj, v_rfno,
  8                          v_bno, v_rno);
  8    rdbms_output.put_line('v_rfno = "'||v_rfno||'"');
  9    dbms_output.put_line('v_bno = "'||v_bno||'"');
 10  end;
 11  /
v_rfno = "1“
v_bno = "180“
```

# RAC contention on a block?

- **Using the same PL/SQL block to translate a ROWID into `file#` and `block#` value…**
  - **We find that all of the sequences previously listed with low CACHE_SIZE values reside within the block at file# = 1 and block# = 180**
- **So, any sequence needing to be refreshed from the less-busy RAC02 instance is going to wait for that block**
  - **Because the more-busy RAC01 instance is dominating access to that block**

# RAC contention on a block?

- **The waits on** “`gc buffer busy`”, “`gc current block 2-way`”, **and** “`gc current request`” **now make more sense**
  1. **We are experiencing high service time for sequence refresh**
  2. **Because a small number of poorly-cached sequences result in lots of modifications to that block**
  3. **Causing the more-busy instance RAC01 to possess the block**
  4. **Leaving the less-busy instance RAC02 having to wait much longer to gain possession of the block**

# Summarizing the findings…

1. Un-cached or poorly-cached sequences are resulting in lots of I/O to one block in the SYS.SEQ$ data dictionary table on the more-busy instance RAC01
2. Resulting in the buffer and row-cache entries for that block to be mastered on the more-busy RAC01
   - Thus waits on the "gc" events…
3. The less-busy RAC02 has to wait to become master of the buffer and row-cache entry for the block before it can even try to update the block
   - Thus waits on event "`enq: SQ – contention`"…
4. This causes contention within the Row Cache of the Shared Pool on RAC02
   - Thus waits on "`row cache lock`" on the section of the Row Cache storing sequence information…

# …to reach a conclusion

- **The resolution for the ORA-03135/3136 turns out to be very simple…**
  - Increase caching on all under-cached sequences
    - Increased sequence caching reduces physical I/O to/from the SYS.SEQ$ table, which reduces inter-instance RAC contention as well (gc waits), which eliminates starvation by lesser-used instance

# Resolution

```
SQL> select   'alter sequence '||sequence_owner||'.'||sequence_name||
  2           ' cache 100 /* prev_cache='||cache_size||',last#='||last_number||' */;'
  3  from     dba_sequences
  4  where    cache_size <= 20
  5  and      (nvl(min_value,0) + last_number) / increment_by >= 10000
  6  order by (nvl(min_value,0) + last_number) / increment_by desc;

CMD
-----------------------------------------------------------------------------------
alter sequence SYS.DBMS_LOCK_ID cache 100    /* prev_cache=20,last#=1073741891 */ ;
alter sequence SYS.IDGEN1$ cache 100         /* prev_cache=20,last#=18987301 */ ;
alter sequence SYS.WRI$_ADV_SEQ_MSGGROUP cache 100 /* prev_cache=10,last#=6428319 */;
alter sequence PROD.JL_SEQ cache 100         /* prev_cache=0,last#=2344665 */ ;
alter sequence SYS.ORA_TQ_BASE$ cache 100    /* prev_cache=0,last#=1895045 */ ;
alter sequence SYS.WRI$_ALERT_SEQUENCE cache 100 /* prev_cache=20,last#=1180544 */ ;
alter sequence MDSYS.TMP_COORD_OPS cache 100 /* prev_cache=0,last#=1000000 */ ;
alter sequence PROD.DL_SEQ cache 100         /* prev_cache=0,last#=964612 */ ;
alter sequence PROD.RUN_ID_SEQ cache 100     /* prev_cache=0,last#=187744 */ ;
alter sequence SYS.SCHEDULER$_INSTANCE_S cache 100 /* prev_cache=20,last#=166467 */ ;
alter sequence SYS.WRI$_ADV_SEQ_TASK cache 100 /* prev_cache=10,last#=151284 */ ;
alter sequence SYS.OBJECT_GRANT cache 100    /* prev_cache20,last#=139402 */ ;
alter sequence PROD.DF_SEQ cache 100         /* prev_cache=0,last#=53402 */ ;
```

# Another case study…

- **Another two-node RAC environment…**
  - **Supporting OBIEE 10.1.3.1.2**
  - **App-server (NQServer.exe) supporting Hyperion users encountering ORA-03135**
    - **Numerous ORA-03136 found frequently in both database instance "alert.log" files**

# Which first seems similar…

- **Similar alert log entries in both instances, at different times…**

```
Mon Oct 31 05:51:07 2011
WARNING: inbound connection timed out (ORA-3136)
Mon Oct 31 05:51:07 2011
WARNING: inbound connection timed out (ORA-3136)
Mon Oct 31 05:51:07 2011
WARNING: inbound connection timed out (ORA-3136)
Mon Oct 31 05:51:07 2011
WARNING: inbound connection timed out (ORA-3136)
Mon Oct 31 05:51:07 2011
WARNING: inbound connection timed out (ORA-3136)
```

# But is quite different…

```
SQL> select   event,
  2           count(*) cnt
  3  from     dba_hist_active_sess_history
  4  where    sample_time between '31-OCT-2011 05:50:30'
  5  and                          '31-OCT-2011 05:51:10'
  6  group by event order by cnt desc;

EVENT                                  CNT  TIME_WAITED
------------------------------ -------- ------------
latch free                             277           81
latch: shared pool                     153           44
latch: library cache                    45           13
reliable message                         1            1
                                        12            0
DBMS_LDAP: LDAP operation                1            0
```

# Research…

- Latch contention…
  - “latch free”
  - “latch: shared pool”
  - “latch: library cache”
- No RAC-related events
  - Nothing to do with sequences or row-cache either…

# Research (cont'd)…

```
SQL> select name, parameter1, parameter2, parameter3
  2  from   v$event_name
  3  where  name in ('latch free','latch: shared pool',
  4                  'latch: library cache');

NAME                    PARAMETER1     PARAMETER2     PARAMETER3
----------------------- -------------- -------------- ------------
latch: shared pool      address        number         tries
latch: library cache    address        number         tries
latch free              address        number         tries
```

# Research (cont’d)…

```
SQL> select   h.event, l.name, count(*) cnt
  2  from     dba_hist_active_sess_history h, v$latchname l
  3  where    h.sample_time between ‘31-OCT-2012 05:50:30'
  4  and                            ‘31-OCT-2012 05:51:10'
  5  and      h.event in ('latch free','latch: shared pool',
  6                       'latch: library cache')
  7  and      l.latch# = h.p2
  8  group by h.event, l.name
  9  order by cnt desc;

EVENT                      NAME                                     CNT
---------------------- ---------------------------------------- --------
latch free                 parameter table allocation management    174
latch: shared pool         shared pool                               153
latch free                 user lock                                  90
latch: library cache       library cache                              45
latch free                 active checkpoint queue latch              12
latch free                 kokc descriptor allocation latch            1
```

# Research (cont'd)…

- `parameter table allocation management`
  - **Automatic SGA management (10.1 → 10.2)**
- `kokc descriptor allocation latch`
  - **Related to high concurrency in XML/XDB (10.2.0.4+)**
- `user lock`
  - **Acquired when cleaning dead user processes or dropping a schema user (from note on MOS)**
- `active checkpoint queue latch`
  - **Database full/incremental checkpoint in progress**
- `shared pool`
  - **Synchronizes changes to Shared Pool in SGA**
- `library cache`
  - **Synchronizes changes to sub-component of Shared Pool (*hard parsing*?)**

# Research summary

- Not RAC-related
- Not sequence-caching related
- Not hard-parsing (i.e. no bind variables)

- Checkpointing?  LGWR problems?
- Process cleanup?  PMON issues?
- Auto SGA management?

# Auto SGA Management

- SGA_TARGET and SGA_MAX_SIZE
  - Both set to 20G
- DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE, JAVA_POOL_SIZE, STREAMS_POOL_SIZE, and LOG_BUFFER all unset
  - displays as "0"
- Has automatic SGA management (ASMM) been particularly active?
  - ...bingo!...

# Auto SGA Management

- Querying the view V$SGA_RESIZE_OPS
  - Sorting by START_TIME
- Showed that:
  - SHRINK of space (i.e. 128 Mb) from the Shared Pool
  - GROWTH of space (i.e. 128 Mb) to the DEFAULT Buffer Cache
  - Both the SHRINK and the GROW operations started on 31-Oct 2011 at 05:38:56 and completed at 05:51:07

**What do you suppose happens to operations within a busy Shared Pool when it is being shrunk?**

# Dampening the thrashing…

- When SGA_TARGET > 0, then the meaning of the standard SGA sizing parameters changes
    - Instead of specifying a static size…
    - …they become a “floor” or minimum value
- So for example, setting SHARED_POOL_SIZE = 6G when SGA_TARGET > 0 means…
    - Shared Pool can still be automatically resized larger than 6G, but not smaller…

# Resolving 2nd case study

- **Dampen the *thrashing* or the constant give-and-tak of space between the Buffer Cache and the Shared Pool**
  - If Buffer Cache is shrunk or constrained, then more cache misses and *physical read I/O* is the result
  - If Shared Pool is shrunk or constrained, then more latch contention is the result
  - Based on past history of auto re-sizing, set a floor value for the Shared Pool

```
ALTER SYSTEM SET SHARED_POOL_SIZE =
3584M;
```

# Summary

- Although primarily regarded as a performance tuning tool
  - ASH is also a great diagnostic tool
    - If you understand how it is populated and its limitations

- Database forensics
  - Understand what is recorded and stored within the database
  - Understand how that information can be used

# Download a free trial of Delphix

Go to http://Delphix.com/ and click the FREE TRIAL button

## Questions?

Email: tim.gorman@delphix.com

Add'l questions: http://community.Delphix.com

Blog: http://EvDBT.com/

Mobile: +1 (303) 885-4526

Thank you so much for listening!

# Please evaluate this session

**Session 2 – RDBMS Forensics: Troubleshooting Using ASH**