Optimizing Oracle 11g Performance

SGA, PGA, Buffer Cache, Keep Pool, Recycle Pool & Result Cache

TidalScale™

Sharding Motherboards Since 2013

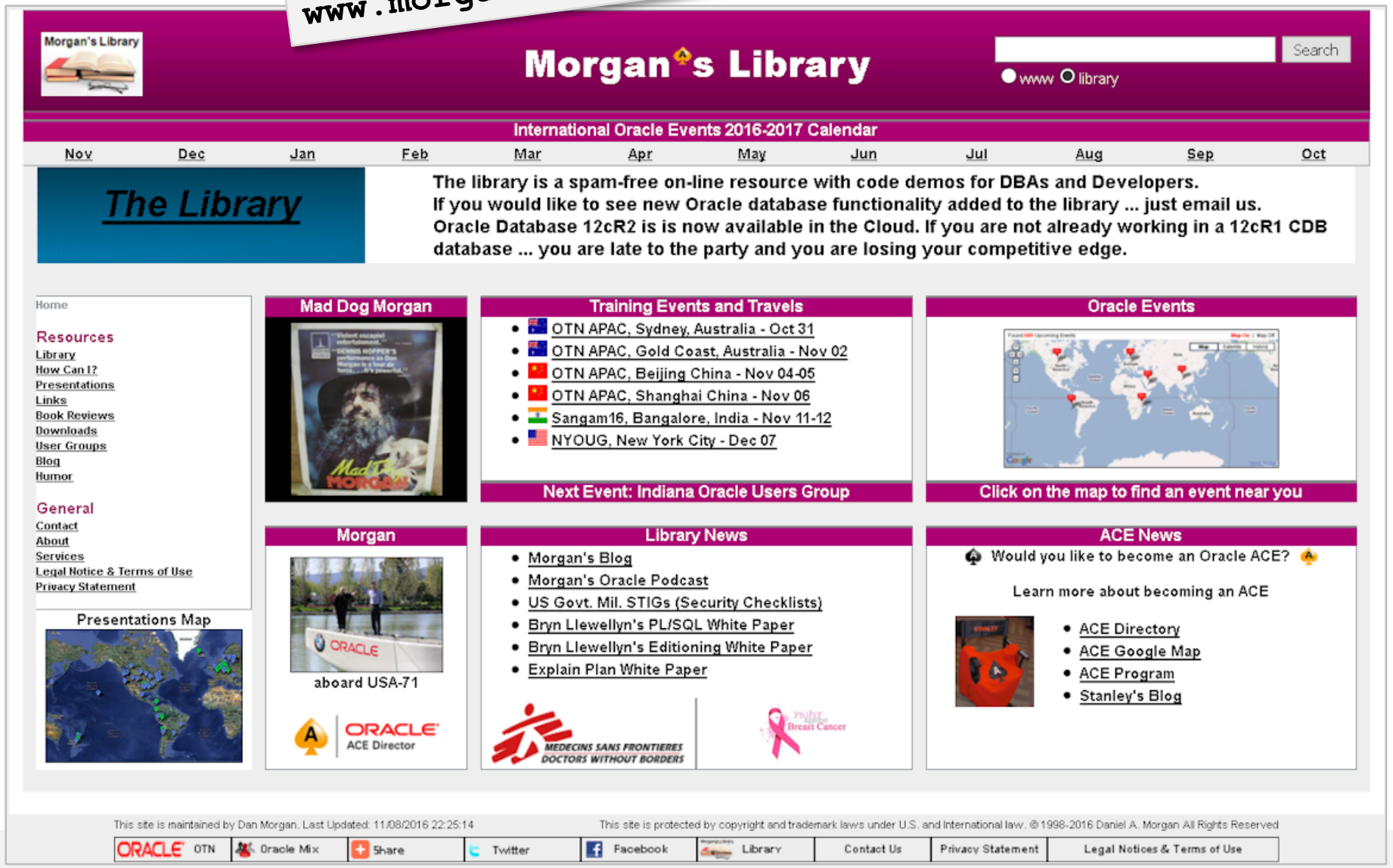
daniel.morgan@tidalscale.com

- Director of Applications @ TidalScale
- Oracle ACE Director Alum
- Oracle Educator
  - Curriculum author and primary instructor, Oracle Program, University of Washington 1998-2009
  - Consultant: Harvard University
    - Guest lecturer at universities in Canada, Chile, Costa Rica, New Zealand, Norway, Panama
    - Frequent lecturer at Oracle conferences … 43 countries since 2008
- IT Professional
  - 2019 will be my 50th year in IT
  - First computer: IBM 360/40 in 1969: Fortran IV
  - Oracle Database since 1988-9 and Oracle Beta tester
  - The Morgan behind www.morganslibrary.org
  - Member Oracle Data Integration Solutions Partner Advisory Council
  - Founding member International TidalScale User Community (ITUC)

TidalScale™

My Personal Website

www.morganslibrary.org



TidalScale™

Performance Problems Have Serious Consequences . . .

- Internal and External customers have expectations
- There is a long history of disappoint
- Thus, we have Service Level Agreements

When We Fail To Deliver . . .

- Internal customers develop their own solutions
- External customers go elsewhere
- SLA violations result in financial penalties
- Management wonders whether we are providing value

Only 2 Things Matter In Business Computing . . .

QoS

- Stability
- Security
- Scalability
- Usability
- Performance

TCO

- Affordability

The History of Oracle Performance Tuning . . .

TidalScale™

How Many Books Read?

How Many Oracle Tools Deployed?

- DBMS_SUPPORT (version 7.2)
- DBMS_TRACE (version 8.1.5)
- DBMS_MONITOR (version 10gR1)
- Oracle Enterprise Manager (OEM)
- StatsPack, ADDM, ASH, AWR, TKPROF, ...

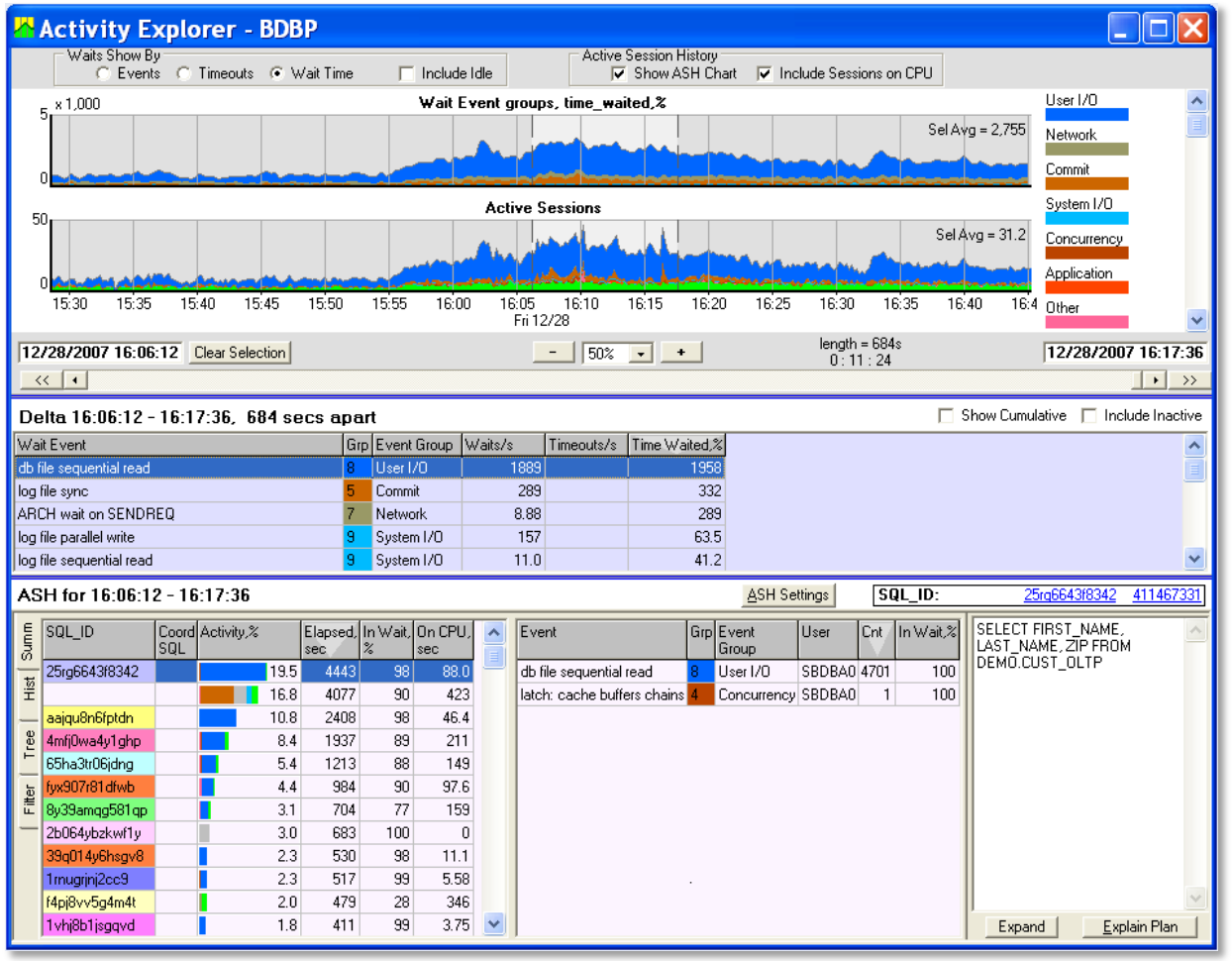
```
ALTER SESSION SET tracefile_identifier = 'test_plan1';

ALTER SESSION SET EVENTS '10053 trace name context forever, level 1';

ALTER SESSION SET EVENTS '10046 trace name context forever, level 12';

-- execute SQL

ALTER SESSION SET EVENTS '10053 trace name context OFF';
ALTER SESSION SET EVENTS '10046 trace name context OFF';
or
ALTER SESSION SET SQL_TRACE=FALSE;

review the trace file in $ORACLE_BASE/diag/orabase/orabase/trace
```

TidalScale™

How Many Tools Have You Purchased?

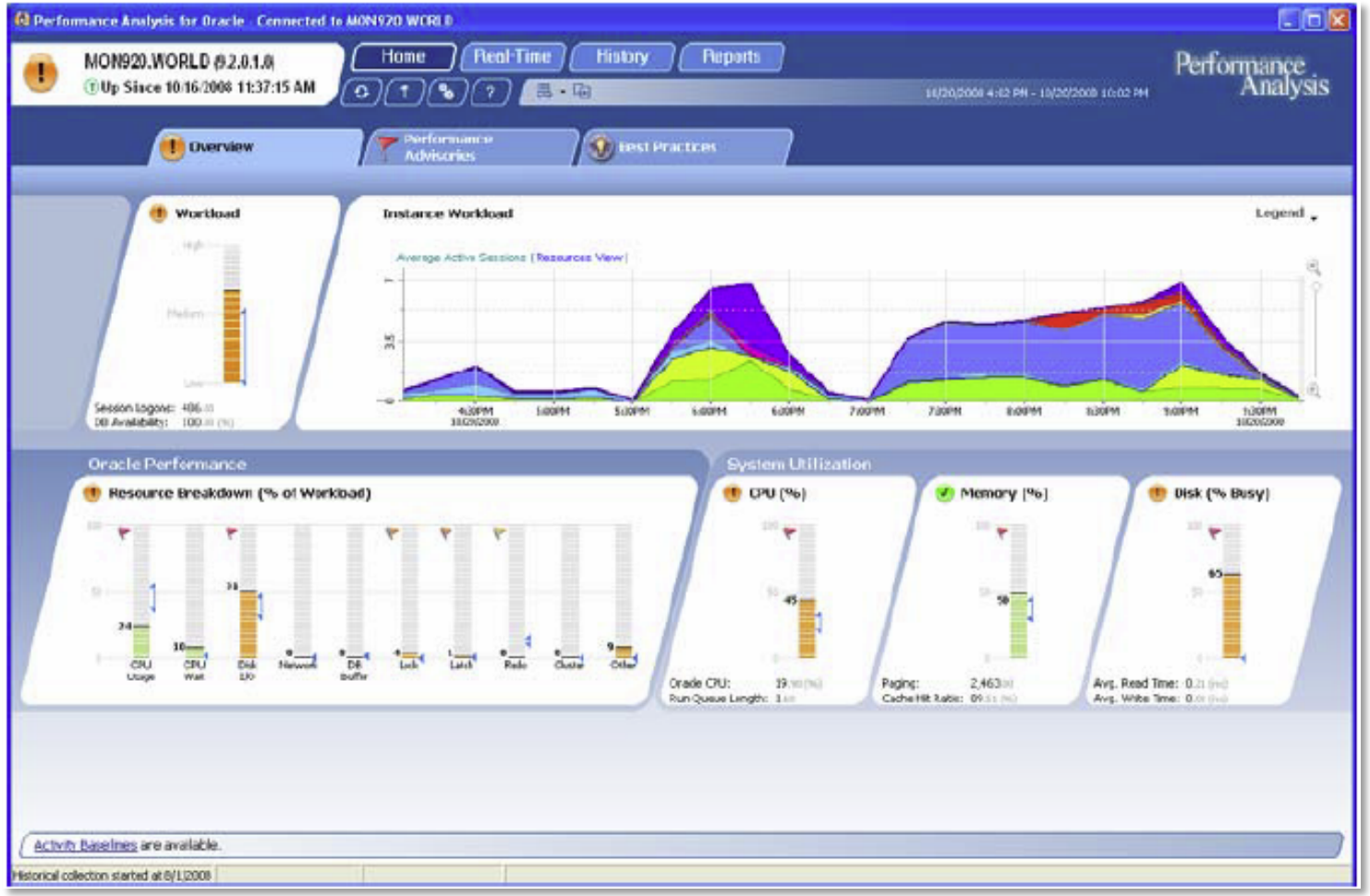
TidalScale™

How Many Tools Have You Purchased?



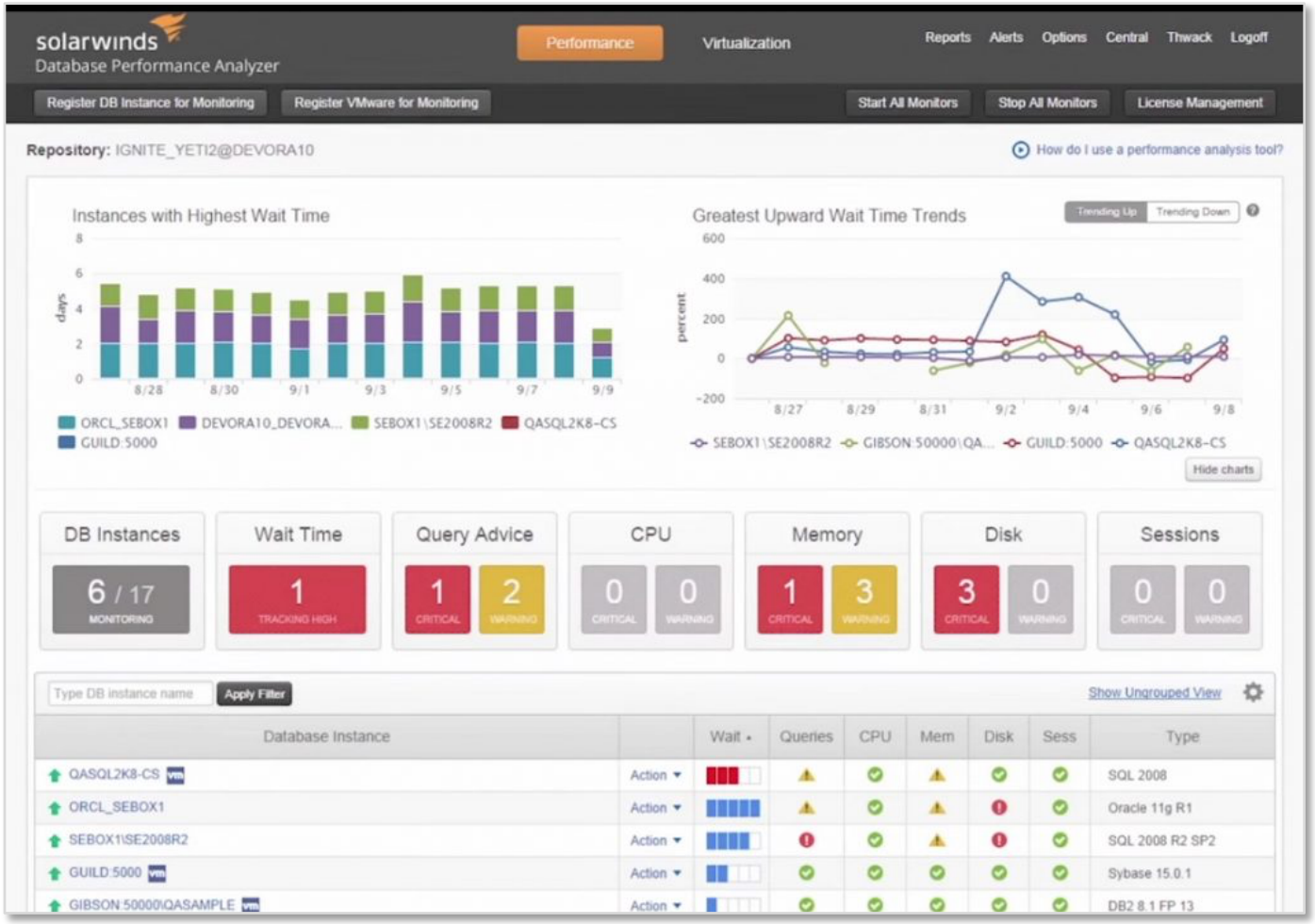
TidalScale™

How Many Tools Have You Purchased?



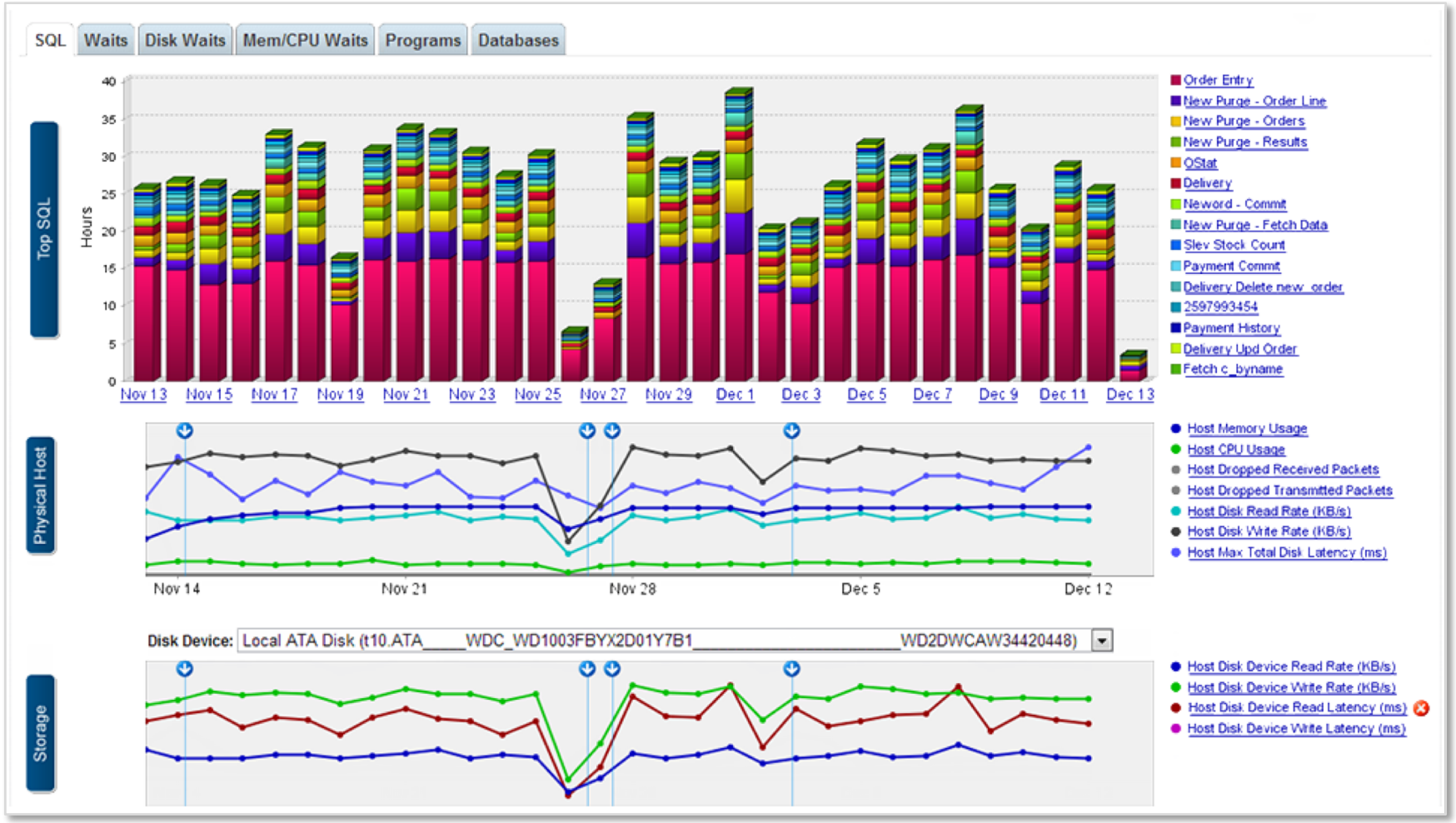
TidalScale™

How Many Tools Have You Purchased?



TidalScale™

How Many Tools Have You Purchased?



TidalScale™

How Many Startup Parameters Configured?

Initialization Parameter	Description
BITMAP_MERGE_AREA_SIZE	Specifies the amount of memory used to merge bitmaps retrieved from an index range scan
DB_BIG_TABLE_CACHE_PERCENT_TARGET	Specifies the cache section target size for automatic big table caching, as a percentage of the buffer cache
DB_nK_CACHE_SIZE	Holds 8K table and index blocks
CREATE_BITMAP_AREA_SIZE	Memory allocated for bitmap creation a larger value may speed up index creation
DB_BLOCK_BUFFERS	Specifies the number of database buffers in the buffer cache
DB_CACHE_SIZE	Specifies the size of the DEFAULT buffer pool for buffers with the primary block size
DB_FLASH_CACHE_SIZE	Specifies the size of the Database Smart Flash Cache
DB_KEEP_CACHE_SIZE	Specifies the size of the KEEP buffer pool
DB_RECYCLE_CACHE_SIZE	Specifies the size of the RECYCLE buffer pool
HASH_AREA_SIZE	Specifies the maximum amount of memory, in bytes, to be used for hash joins
JAVA_MAX_SESSIONSPACE_SIZE	Memory that holds Java state from one database call to another
JAVA_POOL_SIZE	Pool, from which the Java memory manager allocates most Java state during runtime execution
LARGE_POOL_SIZE	Specifies (in bytes) the size of the large pool allocation heap
LOG_BUFFER	Memory used when buffering redo entries to a redo log file
MEMOPTIMIZE_POOL_SIZE	Specifies the size of the memoptimize pool, a memory area in the SGA used by the Memoptimized Rowstore

TidalScale™

Initialization Parameter	Description
MEMORY_MAX_TARGET	Specifies the maximum value to which a DBA can set the MEMORY_TARGET initialization parameter
MEMORY_TARGET	Specifies the Oracle system-wide usable memory
OBJECT_CACHE_MAX_SIZE_PERCENT	specifies the percentage of the optimal cache size that the session object cache can grow past the optimal size
OBJECT_CACHE_OPTIMAL_SIZE	Specifies the size by which the session object cache is reduced when the cache size exceeds the maximum size
OLAP_PAGE_POOL_SIZE	Specifies the size of the OLAP page pool
PGA_AGGREGATE_LIMIT	Specifies a limit on the aggregate PGA memory consumed by the instance
PGA_AGGREGATE_TARGET	Specifies the target aggregate PGA memory available to all server processes attached to the instance
PRE_PAGE_SGA	Specifies whether Oracle reads the entire SGA into memory at startup so that O/S page table entries are pre-built for the SGA
SGA_MAX_SIZE	Specifies the maximum size of the SGA for the lifetime of the instance
SGA_MIN_SIZE	Specifies the minimum size of the SGA for the lifetime of the instance
SGA_TARGET	Specifies the total size of all SGA components
SHARED_POOL_RESERVED_SIZE	Specifies the shared pool space reserved for large contiguous requests for shared pool memory
SHARED_POOL_SIZE	Specifies the size of the shared pool which contains shared cursors, stored procedures, control and other structures
SORT_AREA_RETAINED_SIZE	Specifies the maximum amount of the user global area (UGA) memory retained after a sort run completes
SORT_AREA_SIZE	Specifies the maximum amount of memory Oracle will use for a sort
STREAMS_POOL_SIZE	Specifies the memory allocated for Streams, GoldenGate Integrated Capture and other related processes
USE_LARGE_PAGES	Specify the management of the database's use of large pages for SGA memory

TidalScale™

AWR_report.html

WORKLOAD REPOSITORY report for

DB Name	DB Id	Unique Name	Role	Edition	Release	RAC	CDB
ORCL	1499046141	orcl	PRIMARY	EE	12.2.0.1.0	NO	NO

Instance	Inst Num	Startup Time
oracle	1	25-Aug-18 16:08

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
oracle7002	Linux x86 64-bit	36	36	36	1153.16

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	2713	27-Aug-18 00:46:47	47	.8
End Snap:	2714	27-Aug-18 00:58:57	103	.8
Elapsed:		12.18 (mins)		
DB Time:		138.66 (mins)		

Report Summary

Top ADDM Findings by Average Active Sessions

Finding Name	Avg active sessions of the task	Percent active sessions of finding	Task Name	Begin Snap Time	End Snap Time
Top SQL Statements	11.40	70.40	ADDM:1499046141_1_2714	27-Aug-18 00:46	27-Aug-18 00:58
Undersized PGA	11.40	3.47	ADDM:1499046141_1_2714	27-Aug-18 00:46	27-Aug-18 00:58
Undersized SGA	11.40	2.82	ADDM:1499046141_1_2714	27-Aug-18 00:46	27-Aug-18 00:58
Unusual "Other" Wait Event	11.40	2.27	ADDM:1499046141_1_2714	27-Aug-18 00:46	27-Aug-18 00:58

Memory Statistics

	Begin	End
Host Mem (MB):	1,180,832.7	1,180,832.7
SGA use (MB):	972,800.0	972,800.0
PGA use (MB):	361.9	7,848.7
% Host Mem used for SGA+PGA:	82.41	83.05

Cache Sizes

	Begin	End		
Buffer Cache:	96,768M	96,768M	Std Block Size:	8K
Shared Pool Size:	202,163M	202,149M	Log Buffer:	495,048K
In-Memory Area:	665,600M	665,600M		

Shared Pool Statistics

	Begin	End
Memory Usage %:	3.57	3.66
% SQL with executions>1:	91.10	90.41
% Memory for SQL w/exec>1:	89.90	87.87

Foreground Wait Events

- s - second, ms - millisecond, us - microsecond, ns - nanosecon d
- Only events with Total Wait Time (s) >= .001 are shown
- ordered by wait time desc, waits desc (idle events last)
- %Timeouts: value of 0 indicates value was < .5%. Value of n ull is truly 0

Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait	Waits /txn	% DB time
direct path write temp	25,156		286	11.37ms	613.56	3.44
PGA memory operation	191,325		189	.99ms	4,666.46	2.27
library cache: mutex X	2,415		27	11.24ms	58.90	0.33

TidalScale™

SGA and PGA . . .

- Every DBA on the planet knows their database has an SGA, PGA, and UGA
- Memory management is complex so we let OUI and DBCA figure out what is best
- Then we let ASMM or AMM figure out the rest
- Very few DBAs have the time to review the advisors
- Even fewer regularly run ADDM
- And no one runs an AWR Report unless there is a known issue
- Almost no one reads the Alert Log unless there is an outage

```
2018-10-09T10:49:17.562700-07:00
WARNING: pga_aggregate_limit value is too high for the
amount of physical memory on the system
  PGA_AGGREGATE_LIMIT is 102400 MB
  PGA_AGGREGATE_TARGET is 51200 MB.
  physical memory size is 1180834 MB
  limit based on physical memory and SGA usage is 89950 MB
  SGA_TARGET is 972800 MB
Using default pga_aggregate_limit of 102400 MB
```

TidalScale™

Buffer Cache . . .

- Stores blocks read from disk
- Blocks in the Buffer Cache result in faster Logical I/O instead of Physical I/O
- Configure the buffer cache size with the init parameter DB_BLOCK_BUFFERS
  - The largest size is O/S dependent so read the docs
  - By default the value is zero (0) which means the database manages it

```
ALTER SYSTEM SET db_cache_size=256GB sid="*" scope="BOTH";
```

- To use the database's built-in Advisor you must

```
ALTER SYSTEM SET db_cache_advice=ON sid="*" scope="BOTH";
```

- Then you can

```
SELECT * FROM v$dbcache_advice;
```

```
SQL> SELECT DISTINCT view_name FROM cdb_views
  2  WHERE view_name LIKE '%BUFFER_POOL%';

VIEW_NAME
------------------------------
CDB_HIST_BUFFER_POOL_STAT
DBA_HIST_BUFFER_POOL_STAT
V_$BUFFER_POOL
V_$BUFFER_POOL_STATISTICS
```

TidalScale™

Buffer Cache . . .

	Begin	End
Host Mem (MB):	515,233.3	515,233.3
SGA use (MB):	49,969.1	49,969.1
PGA use (MB):	13,351.0	9,493.1
% Host Mem used for SGA+PGA:	12.29	12.29

	Begin	End		
Buffer Cache:	25,600M	25,600M	Std Block Size:	32K
Shared Pool Size:	20,480M	20,480M	Log Buffer:	310,312K

About the Database Buffer Cache

For many types of operations, Oracle Database uses the buffer cache to store data blocks read from disk. Oracle Database bypasses the buffer cache for particular operations, such as sorting and parallel reads.

To use the database buffer cache effectively, tune SQL statements for the application to avoid unnecessary resource consumption. To meet this goal, verify that frequently executed SQL statements and SQL statements that perform many buffer gets are well-tuned.

When using parallel query, consider configuring the database to use the database buffer cache instead of performing direct reads into the Program Global Area (PGA). This configuration may be appropriate when the system has a large amount of memory.

TidalScale™

OpenWorld Benchmark: Run 1

- 2 node TidalPod, 1.2TB Linux

```
[tsadmin@oracle7002 ~]$ free
              total        used        free      shared  buff/cache   available
Mem:     1209174368     8912196     5558776   734039944  1194703396   462374144
Swap:            0           0           0
```

- Default Oracle installation (./runInstaller, next, next, next, next, next, ...) ver. 12cR2

```
SQL> show parameter sga

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
allow_group_access_to_sga            boolean     FALSE
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     TRUE
sga_max_size                         big integer 356864M
sga_min_size                         big integer 0
sga_target                           big integer 356864M
unified_audit_sga_queue_size         integer     1048576
```

```
SQL> show parameter buffer

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------
buffer_pool_keep                     string
buffer_pool_recycle                  string
db_block_buffers                     integer     0
log_buffer                           big integer 483352K
```

```
SQL> show parameter pga

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------
pga_aggregate_limit                  big integer 237794M
pga_aggregate_target                 big integer 118897M
```

```
SQL> show parameter workarea

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
workarea_size_policy                 string      AUTO
```

- OUI/DBCA is designed for the typically memory starved deployment

TidalScale™

OpenWorld Benchmark: Run 1

- Linux presents 1.2TB of memory ... Note how Oracle sizes itself
- The SGA is only 374G
- The Buffer Cache is only 328G
- Does anyone need a 3.2G Java Cache?

- Oracle's automatic memory management assumes you are memory starved
  If you install Oracle onto a server with a large amount of memory it gives the majority of it to Linux not the database

```
SQL> SELECT * FROM v$sga;

NAME                             VALUE
-------------------------------- -------------
Fixed Size                         26344872
Variable Size                   45634028120
Database Buffers                328028127232
Redo Buffers                      510525440

SQL> SELECT * FROM v$sgainfo;

NAME                           BYTES      RES
------------------------------ ---------- ---
Fixed SGA Size                   26344872 No
Redo Buffers                    510525440 No
Buffer Cache Size              3.2803E+11 Yes
In-Memory Area Size                     0 No
Shared Pool Size               4.0802E+10 Yes
Large Pool Size                1610612736 Yes
Java Pool Size                 3221225472 Yes
Streams Pool Size                       0 Yes
Shared IO Pool Size             536870912 Yes
Data Transfer Cache Size                0 Yes
Granule Size                    536870912 No
Maximum SGA Size               3.7420E+11 No
Startup overhead in Shared Pool 3541291328 No
```

TidalScale™

OpenWorld Benchmark: Run 1

- Monitor long running transaction (> 7 seconds) with V$SESSION_LONGOPS

```
SQL> SELECT target, sofar, totalwork, units, elapsed_seconds, message
  2  FROM v$session_longops
  3  ORDER BY start_time;


TARGET               SOFAR TOTALWORK UNITS   ELAPSED_SECONDS MESSAGE
-------------- ---------- --------- ------- --------------- --------------------------------------------------------------------
TPCH.LINEITEM    52682157   52720780 Blocks             1771 Table Scan: TPCH.LINEITEM: 52682157 out of 52720780 Blocks done

SQL> /

TARGET               SOFAR TOTALWORK UNITS   ELAPSED_SECONDS MESSAGE
-------------- ---------- --------- ------- --------------- --------------------------------------------------------------------
TPCH.LINEITEM    52720780   52720780 Blocks             1772 Table Scan: TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
TPCH.PART          996684    1922753 Blocks              190 Table Scan: TPCH.PART: 996684 out of 1922753 Blocks done
```

TidalScale™

OpenWorld Benchmark: Run 1

```
SQL> /

TARGET              SOFAR TOTALWORK UNITS   ELAPSED_SECONDS TIME_REMAINING MESSAGE
-------------- ---------- --------- ------- --------------- -------------- ----------------------------------------------------------------
TPCH.LINEITEM    52720780  52720780 Blocks             1772              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
TPCH.PART         1922753   1922753 Blocks              349              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
                    95480     95480 Blocks                9              0 Hash Join: : 95480 out of 95480 Blocks done
TPCH.PART         1922753   1922753 Blocks              516              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              737              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
TPCH.SUPPLIER       95777     95777 Blocks               17              0 Table Scan:  TPCH.SUPPLIER: 95777 out of 95777 Blocks done
TPCH.PART         1922753   1922753 Blocks              160              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              468              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
TPCH.LINEITEM    52720780  52720780 Blocks             1748              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
                 17327853  17327853 Blocks              954              0 Hash Join: : 17327853 out of 17327853 Blocks done
TPCH.ORDERS      11571678  11571678 Blocks              688              0 Table Scan:  TPCH.ORDERS: 11571678 out of 11571678 Blocks done
                   758268    758268 Blocks             1009              0 Sort Output: : 758268 out of 758268 Blocks done
                  2213400   2213400 Blocks              123              0 Sort/Merge: : 2213400 out of 2213400 Blocks done
TPCH.LINEITEM    52720780  52720780 Blocks             1520              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              172              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
                  1190400   1190400 Blocks               62              0 Sort/Merge:  : 1190400 out of 1190400 Blocks done
TPCH.PART         1922753   1922753 Blocks              124              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
                  1183339   1190389 Blocks              154              1 Sort Output:  : 1183339 out of 1190389 Blocks done
TPCH.SUPPLIER       95777     95777 Blocks               21              0 Table Scan:  TPCH.SUPPLIER: 95777 out of 95777 Blocks done
TPCH.LINEITEM     6230213  52720780 Blocks               88            657 Table Scan:  TPCH.LINEITEM: 6230213 out of 52720780 Blocks done
```

- SORT_AREA_SIZE, by default, is 64K
- Hash Joins take place in a pool specified as (2 x SORT_AREA_SIZE)
- A 782M Hash Join was took place in a 128K cache forcing swapping to disk
- Sort/Merge + Sort Output is 43.85G ... requiring substantial memory resources

TidalScale™

OpenWorld Benchmark: Run 1

```
SQL> /

TARGET               SOFAR TOTALWORK UNITS   ELAPSED_SECONDS TIME_REMAINING MESSAGE
-------------- ---------- --------- ------- --------------- -------------- ------------------------------------------------------------------
TPCH.LINEITEM    52720780  52720780 Blocks             1772              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
TPCH.PART         1922753   1922753 Blocks              349              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
                    95480     95480 Blocks                9              0 Hash Join:  : 95480 out of 95480 Blocks done
TPCH.PART         1922753   1922753 Blocks              516              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              737              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
TPCH.SUPPLIER       95777     95777 Blocks               17              0 Table Scan:  TPCH.SUPPLIER: 95777 out of 95777 Blocks done
TPCH.PART         1922753   1922753 Blocks              160              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              468              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
TPCH.LINEITEM    52720780  52720780 Blocks             1748              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
                 17327853  17327853 Blocks              954              0 Hash Join:  : 17327853 out of 17327853 Blocks done
TPCH.ORDERS      11571678  11571678 Blocks              688              0 Table Scan:  TPCH.ORDERS: 11571678 out of 11571678 Blocks done
                   758268    758268 Blocks             1009              0 Sort Output:  : 758268 out of 758268 Blocks done
                  2213400   2213400 Blocks              123              0 Sort/Merge:  : 2213400 out of 2213400 Blocks done
TPCH.LINEITEM    52720780  52720780 Blocks             1520              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              172              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
                  1190400   1190400 Blocks               62              0 Sort/Merge:  : 1190400 out of 1190400 Blocks done
TPCH.PART         1922753   1922753 Blocks              124              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
                  1183339   1190389 Blocks              154              1 Sort Output:  : 1183339 out of 1190389 Blocks done
TPCH.SUPPLIER       95777     95777 Blocks               21              0 Table Scan:  TPCH.SUPPLIER: 95777 out of 95777 Blocks done
TPCH.LINEITEM     6230213  52720780 Blocks               88            657 Table Scan:  TPCH.LINEITEM: 6230213 out of 52720780 Blocks done
```

- Having rows in the buffer cache really matters (88 + 657 = 745 seconds)
- Just as building storage indexes on an Exadata improves performance
- If you don't have enough memory you cannot stop LRU from flushing your data

TidalScale™

OpenWorld Benchmark: Run 1

```
SQL> /

TARGET               SOFAR TOTALWORK UNITS   ELAPSED_SECONDS TIME_REMAINING MESSAGE
-------------- ---------- --------- ------- --------------- -------------- ------------------------------------------------------------------------
TPCH.LINEITEM    52720780  52720780 Blocks             1772              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
TPCH.PART         1922753   1922753 Blocks              349              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
                    95480     95480 Blocks                9              0 Hash Join:  : 95480 out of 95480 Blocks done
TPCH.PART         1922753   1922753 Blocks              516              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              737              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
TPCH.SUPPLIER       95777     95777 Blocks               17              0 Table Scan:  TPCH.SUPPLIER: 95777 out of 95777 Blocks done
TPCH.PART         1922753   1922753 Blocks              160              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              468              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
TPCH.LINEITEM    52720780  52720780 Blocks             1748              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
                 17327853  17327853 Blocks              954              0 Hash Join:  : 17327853 out of 17327853 Blocks done
TPCH.ORDERS      11571678  11571678 Blocks              688              0 Table Scan:  TPCH.ORDERS: 11571678 out of 11571678 Blocks done
                   758268    758268 Blocks             1009              0 Sort Output:  : 758268 out of 758268 Blocks done
                  2213400   2213400 Blocks              123              0 Sort/Merge:  : 2213400 out of 2213400 Blocks done
TPCH.LINEITEM    52720780  52720780 Blocks             1520              0 Table Scan:  TPCH.LINEITEM: 52720780 out of 52720780 Blocks done
TPCH.PARTSUPP     8445394   8445394 Blocks              172              0 Table Scan:  TPCH.PARTSUPP: 8445394 out of 8445394 Blocks done
                  1190400   1190400 Blocks               62              0 Sort/Merge:  : 1190400 out of 1190400 Blocks done
TPCH.PART         1922753   1922753 Blocks              124              0 Table Scan:  TPCH.PART: 1922753 out of 1922753 Blocks done
                  1183339   1190389 Blocks              154              1 Sort Output:  : 1183339 out of 1190389 Blocks done
TPCH.SUPPLIER       95777     95777 Blocks               21              0 Table Scan:  TPCH.SUPPLIER: 95777 out of 95777 Blocks done
TPCH.LINEITEM     6230213  52720780 Blocks               88            657 Table Scan:  TPCH.LINEITEM: 6230213 out of 52720780 Blocks done
```

- Having rows in the buffer cache improves performance
- Just as building storage indexes on an Exadata improves performance
- If you don't have enough memory you cannot stop LRU from flushing your data

TidalScale™

OpenWorld Benchmark: Run 1

- Each memory resize requires latching and serialization
- The more resize operations the slower performance

```
SQL> SELECT component, parameter, start_time, end_time, initial_size, target_size, final_size
  2  FROM v$memory_resize_ops
  3* WHERE initial_size + target_size + final_size <> 0;

COMPONENT             PARAMETER             START_TIME          END_TIME            INITIAL_SIZE TARGET_SIZE FINAL_SIZE
--------------------- --------------------- ------------------- ------------------- ------------ ----------- ----------
shared pool           shared_pool_size      05-OCT-2018 08:14:29 05-OCT-2018 08:14:29            0   503316480  503316480
large pool            large_pool_size       05-OCT-2018 08:14:29 05-OCT-2018 08:14:29            0   150994944  150994944
SGA Target            sga_target            05-OCT-2018 08:14:29 05-OCT-2018 08:14:29            0  2550136832 2550136832
DEFAULT buffer cache  db_cache_size         05-OCT-2018 08:14:29 05-OCT-2018 08:14:29   1862270976  1862270976 1862270976
java pool             java_pool_size        05-OCT-2018 08:14:29 05-OCT-2018 08:14:29            0    16777216   16777216
PGA Target            pga_aggregate_target  05-OCT-2018 08:14:29 05-OCT-2018 08:14:29            0   855638016  855638016
DEFAULT buffer cache  db_cache_size         05-OCT-2018 08:14:29 05-OCT-2018 08:14:29            0  1862270976 1862270976
DEFAULT buffer cache  db_cache_size         05-OCT-2018 08:15:28 05-OCT-2018 08:15:29   1862270976  1979711488 1979711488
large pool            large_pool_size       05-OCT-2018 08:15:28 05-OCT-2018 08:15:29    150994944    33554432   33554432x
```

TidalScale™

Keep and Recycle Pools . . .

- The Keep Pool is part of the SGA and is just another Buffer Cache, so is the Buffer Recycle Pool both of which can be used to assign frequently accessed segments
- All of these pools use an LRU algorithm so blocks can age out
- Tom Kyte recommended not using them and we do too
- Too complex without corresponding value

```
CREATE TABLE t1 (
my_date    DATE           NOT NULL,
my_number NUMBER(12,10) NOT NULL,
my_row     NUMBER(12)     NOT NULL)
STORAGE (BUFFER_POOL KEEP);

CREATE UNIQUE INDEX t1_ind1
ON t1(my_date)
STORAGE (BUFFER_POOL KEEP);
```

```
CREATE TABLE t1_r (
my_date    DATE           NOT NULL,
my_number NUMBER(12,10) NOT NULL,
my_row     NUMBER(12)     NOT NULL)
STORAGE (BUFFER_POOL RECYCLE);

CREATE UNIQUE INDEX t1_ind1_R
ON t1_r(my_date)
STORAGE (BUFFER_POOL RECYCLE);
```

```
ALTER SYSTEM SET DB_KEEP_CACHE_SIZE = 16777216 SCOPE='BOTH';
ALTER SYSTEM SET DB_RECYCLE_CACHE_SIZE = 16777216 SCOPE='BOTH';
```

TidalScale™

Result Cache . . . (introduced in 11.1.0.6)

SQL ordered by Executions

- Total Executions: 29,717,627
- Captured SQL account for 77.4% of Total

Executions	Rows Processed	Rows per Exec	CPU per Exec (s)	Elap per Exec (s)	SQL Id	SQL Module	SQL Text
10,128,178	2,506,529	0.25	0.00	0.00	932srzg1krc33	ASN_07B_DIP(004110016)	SELECT NE_TIMEZONE FROM CMPM.E...
7,576,759	7,579,197	1.00	0.00	0.00	1h698sb62un99	asci_56_RANAPProtocolStats(01611000E)	SELECT DISTINCT NE_TIMEZONE FR...
3,914,621	3,848,268	0.98	0.00	0.00	5tbzddgguu8cc	asci_56_RANAPProtocolStats(01611000E)	SELECT SYS_VERSION FROM CMPM.T...
311,645	311,604	1.00	0.00	0.00	7gtztzv329wg0		select c.name, u.name from co...
301,428	301,325	1.00	0.00	0.00	36s446f9cnwhw		SELECT C.NAME FROM COL$ C WHER...
200,692	200,669	1.00	0.00	0.00	4vs91dcv7u1p6	OMS	insert into sys.aud$(sessioni...
65,044	65,035	1.00	0.00	0.00	fz9xwpt2cvt0k		SELECT par_type, param_clob, ...
64,949	3,945,482	60.75	0.00	0.00	f5ra7dru5fk5n	XML_P7R_RNC_RCS(003110008)	SELECT NAME, PATH, READ, WR...
64,801	64,807	1.00	0.00	0.00	fhzj09a7fnrnb	XML_V7I_IN_LP_DC(00811000V)	SELECT DBTIMEZONE, LENGTH(DBT...
64,632	64,542	1.00	0.00	0.00	15jnrrb6016nd	XML_V7I_IN_LP_DC(00811000V)	SELECT SESSIONTIMEZONE, LENGT...

```
10.1M executions of SELECT timezone
 7.6M executions of SELECT DISTINCT timezone
 3.9M executions of SELECT version

How many times in an hour does your server change its time zone?
How many times in an hour does your software change version number?
```

TidalScale™

Wrap Up . . .

- Memory is 300X faster than flash
- The more of your data you can cache in memory the more your performance will improve due to reducing PIO and increasing LIO
- Oracle's installation tools have been written to expect insufficient memory
- If you have provided your database with sufficient memory you manually size
  - SGA, PGA, and Result Cache
  - Use ASMM for automatic memory management
- With Software Defined Servers you can create an environment with the amount of cpu and memory required to
  - Improve performance
  - Control your licensing cost

TidalScale™

Next Steps

Contact me directly to

- Answer questions about TidalScale Software Defined Servers
- Present TidalScale Software Defined Servers to your team
- Identify opportunities in your organization for Software Defined Servers

daniel.morgan@tidalscale.com

TidalScale™