TidalScale

Software-Defined Servers

TidalScale™

TidalScale

Faster Results – Lower Cost

Software-Defined Servers

TidalScale™

Unsafe Harbor Statement

- This room is an unsafe harbor
- You can rely on the information in this presentation to help you improve the performance of your Oracle software and your career
- Everything I will present is existing, proven functionality

- Only 2 things matter in computing ... QoS and TCO

Agenda

- Introduction
- Performance Tuning 101
- What Doesn’t Work
- What Does: Software Defined Servers
- Wrap Up

Introduction

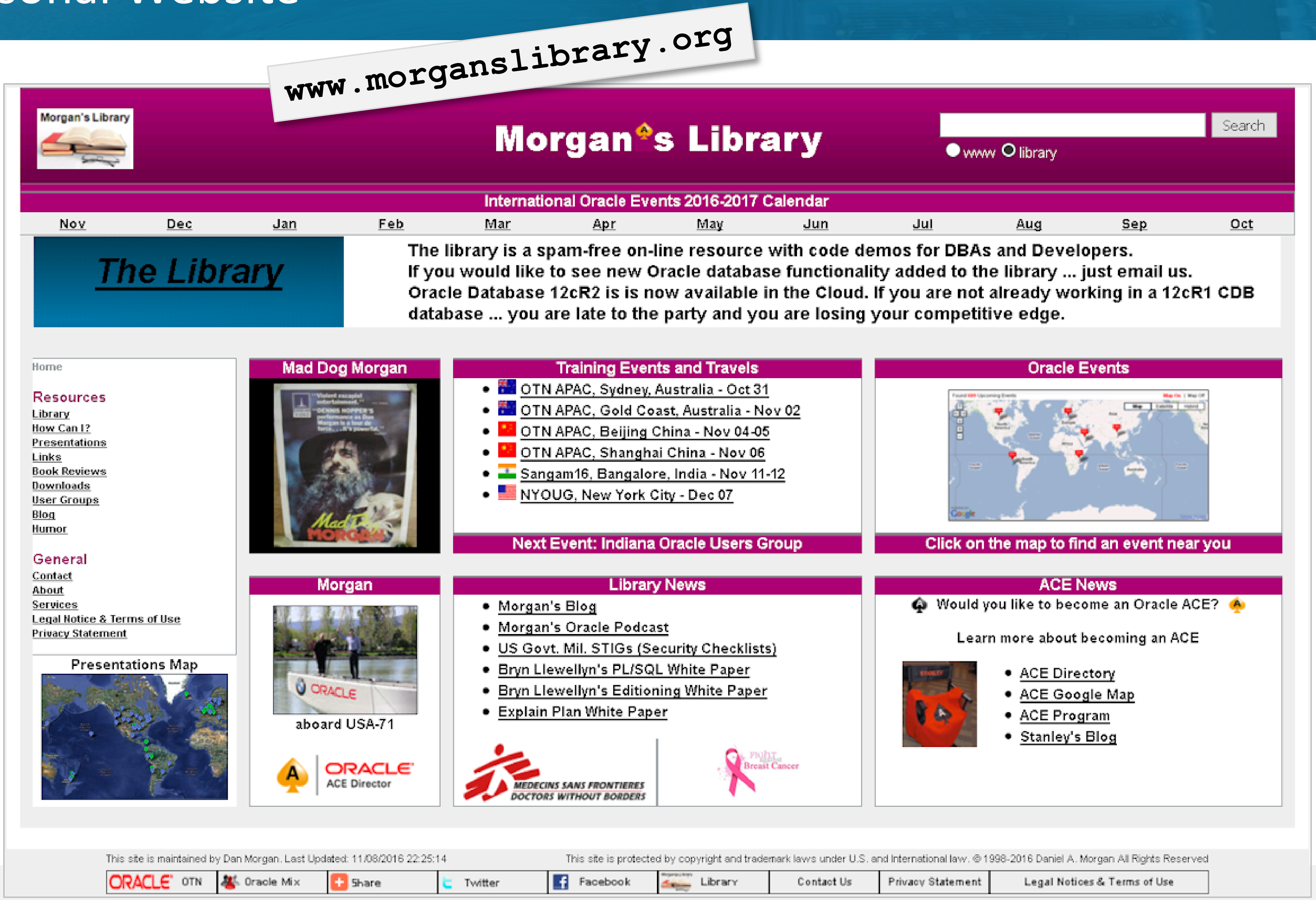
TidalScale™

daniel.morgan@tidalscale.com

- Director of Applications @ TidalScale
- Oracle ACE Director Alumni
- 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 50^{th} 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)

System/370-145 system console

My Personal Website



TidalScale™

TidalScale Snapshot

- Founded in 2013
- Cohesive team with deep data center experience
- Focused on revolutionizing the data center
- Available now: on-prem or Cloud deployments
- 22+ patents approved or pending
- Strong portfolio of investors

Who Is TidalScale?

- Leadership team has helped shape the IT landscape
- We have designed, developed and deployed some of the most important and successful systems and services in the history of the computing industry - internet, Ethernet, operating systems, programming languages and microprocessors
- Elite team has collectively earned dozens of patents, three film credits and grown record setting businesses
- Collectively, we've shipped more than 2 billion licensed products
- Passionate difference makers with a reputation for delivering
- We know how to work hard and still have a life

TidalScale Leadership Team DNA

Dr. Ike Nassi
Founder / CTO

Gary Smerdon
President & CEO

Dr. David Reed
Chief Scientist

Michael Berman
VP Engineering

Dave Ferretti
VP Sales

John Williams
VP Product

Chuck Piercey
VP Marketing

Pete Jarvis
VP Business Development

TidalScale™

Tidal Scale Industry Momentum

Partners & Customers

SAP® Partner Open Ecosystem
SUSE
IBM Bluemix™
TIBCO®
redhat
infor
Infosys
orionVM
ORACLE
ubuntu®

Recognition

Gartner Cool Vendor 2017

IDC Innovator 2017

eWEEK

"TidalScale may have come up with the biggest advance in servers since VMware 18 years ago."

RED HERRING 100 WINNER N. AMERICA

"This is the way all servers will be built in the future"

Gordon Bell, industry legend and 1st investor in TidalScale

TidalScale

Questions and Answers

- There is a lot of new information here: Take notes

- You may interrupt my presentation at any time if you have a question
- I will be available for additional Q&A after this presentation

Performance Tuning 101

Some Times You Need To Rewrite SQL

```
--------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                       | Name                       | Rows  | Bytes |TempSpc| Cost  (%CPU)| Time     |
--------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                                |                            |       |       |       |  264T(100)|          |
|*  1 |  VIEW                                           |                            |  156P|   15E|       |  264T  (79)|999:59:59 |
|*  2 |   WINDOW SORT PUSHED RANK                       |                            |  156P|   15E|   15E|  264T  (79)|999:59:59 |
|   3 |    MERGE JOIN CARTESIAN                         |                            |  156P|   15E|       |   68T  (16)|999:59:59 |
|   4 |     MERGE JOIN CARTESIAN                        |                            |  220G|  205T|       |   96M  (16)| 26:57:48 |
|   5 |      MERGE JOIN CARTESIAN                       |                            |  310K|  302M|       |  232   (11)| 00:00:01 |
|   6 |       MERGE JOIN CARTESIAN                      |                            |  779 |  777K|       |   22    (0)| 00:00:01 |
|   7 |        NESTED LOOPS                             |                            |       |       |       |            |          |
|   8 |         NESTED LOOPS                            |                            |     2 |  2044 |       |   20    (0)| 00:00:01 |
|   9 |          NESTED LOOPS OUTER                     |                            |     2 |  1990 |       |   18    (0)| 00:00:01 |
|  10 |           NESTED LOOPS                          |                            |     2 |  1868 |       |   17    (0)| 00:00:01 |
|  11 |            NESTED LOOPS                         |                            |     2 |  1712 |       |   15    (0)| 00:00:01 |
|  12 |             NESTED LOOPS                        |                            |     2 |  1564 |       |   13    (0)| 00:00:01 |
|  13 |              MERGE JOIN CARTESIAN               |                            |     2 |  1442 |       |   11    (0)| 00:00:01 |
|  14 |               NESTED LOOPS OUTER                |                            |     1 |   625 |       |    8    (0)| 00:00:01 |
|  15 |                NESTED LOOPS OUTER               |                            |     1 |   613 |       |    7    (0)| 00:00:01 |
|  16 |                 NESTED LOOPS                    |                            |     1 |   580 |       |    6    (0)| 00:00:01 |
|  17 |                  NESTED LOOPS OUTER             |                            |     1 |   539 |       |    5    (0)| 00:00:01 |
|  18 |                   NESTED LOOPS OUTER            |                            |     1 |   340 |       |    5    (0)| 00:00:01 |
|  19 |                    TABLE ACCESS BY INDEX ROWID| PA_STUDENT                 |     1 |   316 |       |    3    (0)| 00:00:01 |
|* 20 |                     INDEX UNIQUE SCAN           | PK_STUDENT                 |     1 |       |       |    2    (0)| 00:00:01 |
|  21 |                    TABLE ACCESS BY INDEX ROWID| PA_STUD_USER               |     1 |    24 |       |    2    (0)| 00:00:01 |
|* 22 |                     INDEX UNIQUE SCAN           | PK_STUD_USER               |     1 |       |       |    1    (0)| 00:00:01 |
|  23 |                   TABLE ACCESS BY INDEX ROWID  | PA_ORG                     |     1 |   199 |       |    0    (0)|          |
|* 24 |                    INDEX UNIQUE SCAN            | PK_ORG                     |     1 |       |       |    0    (0)|          |
|  25 |                  TABLE ACCESS BY INDEX ROWID    | PA_DOMAIN                  |    13 |   533 |       |    1    (0)| 00:00:01 |
|* 26 |                   INDEX UNIQUE SCAN             | PK_DOMAIN                  |     1 |       |       |    0    (0)|          |
|  27 |                 TABLE ACCESS BY INDEX ROWID     | PA_USRRF_STUD              |   100 |  3300 |       |    1    (0)| 00:00:01 |
|* 28 |                  INDEX UNIQUE SCAN              | PK_USRRF_STUD              |     1 |       |       |    0    (0)|          |
|  29 |                VIEW PUSHED PREDICATE            | PV_STUD_USER               |     1 |    12 |       |    1    (0)| 00:00:01 |
|* 30 |                 FILTER                          |                            |       |       |       |            |          |
|  31 |                  NESTED LOOPS OUTER             |                            |     1 |    22 |       |  264   (11)| 00:00:01 |
|* 32 |                   INDEX UNIQUE SCAN             | PK_STUDENT                 |     1 |    10 |       |    2    (0)| 00:00:01 |
|* 33 |                   MAT_VIEW ACCESS FULL          | PV_AP_STUD_USER            |     1 |    12 |       |  262   (11)| 00:00:01 |
|  34 |               BUFFER SORT                       |                            |     2 |   192 |       |   10    (0)| 00:00:01 |
|  35 |                TABLE ACCESS BY INDEX ROWID      | PA_CPNT_COMPLIANCE_DATA    |     2 |   192 |       |    3    (0)| 00:00:01 |
|* 36 |                 INDEX RANGE SCAN                | IX_CPNT_CD__EVTHST         |     2 |       |       |    1    (0)| 00:00:01 |
|  37 |              TABLE ACCESS BY INDEX ROWID        | PA_CPNT_TYPE               |     1 |    61 |       |    1    (0)| 00:00:01 |
|* 38 |               INDEX UNIQUE SCAN                 | PK_CPNT_TYPE               |     1 |       |       |    0    (0)|          |
|  39 |             TABLE ACCESS BY INDEX ROWID         | PA_RQMT_TYPE               |     1 |    74 |       |    1    (0)| 00:00:01 |
|* 40 |              INDEX UNIQUE SCAN                  | PK_RQMT_TYPE               |     1 |       |       |    0    (0)|          |
|  41 |            TABLE ACCESS BY INDEX ROWID          | PA_CMPL_STAT               |     1 |    78 |       |    1    (0)| 00:00:01 |
|* 42 |             INDEX UNIQUE SCAN                   | PK_CMPL_STAT               |     1 |       |       |    0    (0)|          |
|  43 |           TABLE ACCESS BY INDEX ROWID           | PA_QUAL                    |     1 |    61 |       |    1    (0)| 00:00:01 |
|* 44 |            INDEX UNIQUE SCAN                    | PK_QUAL                    |     1 |       |       |    0    (0)|          |
|* 45 |          INDEX UNIQUE SCAN                      | PK_CPNT                    |     1 |       |       |    0    (0)|          |
|  46 |         TABLE ACCESS BY INDEX ROWID             | PA_CPNT                    |     1 |    27 |       |    1    (0)| 00:00:01 |
|  47 |        BUFFER SORT                              |                            |   399 |       |       |   21    (0)| 00:00:01 |
|  48 |         INDEX FAST FULL SCAN                    | PK_USRRF_STUD              |   399 |       |       |    1    (0)| 00:00:01 |
|  49 |       BUFFER SORT                               |                            |   399 |       |       |  231   (11)| 00:00:01 |
|  50 |        INDEX FAST FULL SCAN                     | PK_USRRF_STUD              |   399 |       |       |    0    (0)|          |
|  51 |      BUFFER SORT                                |                            |  710K|       |       |   96M  (16)| 26:57:48 |
|  52 |       INDEX FAST FULL SCAN                      | IX_STUD_USER__STUDENT      |  710K|       |       |  309   (16)| 00:00:01 |
|  53 |     BUFFER SORT                                 |                            |  710K|       |       |  264T  (79)|999:59:59 |
|  54 |      INDEX FAST FULL SCAN                       | IX_STUD_USER__STUDENT      |  710K|       |       |  309   (16)| 00:00:01 |
--------------------------------------------------------------------------------------------------------------------------------
```

TidalScale™

Some Times You Need To Rewrite SQL

```
-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name                  |  Rows  | Bytes |TempSpc| Cost (%CPU)|   Time    |
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                       |        |       |       |  264T(100)|           |
|*  1 |  VIEW                                  |                       |  156P|    15E|       |  264T  (79)|999:59:59 |
|*  2 |   WINDOW SORT PUSHED RANK              |                       |  156P|    15E|    15E|  264T  (79)|999:59:59 |
|   3 |    MERGE JOIN CARTESIAN                |                       |  156P|    15E|       |   68T  (16)|999:59:59 |
|   4 |     MERGE JOIN CARTESIAN               |                       |  220G|   205T|       |   96M  (16)| 26:57:48 |
|   5 |      MERGE JOIN CARTESIAN              |                       |  310K|   302M|       |  232   (11)| 00:00:01 |
|   6 |       MERGE JOIN CARTESIAN             |                       |  779 |   777K|       |   22    (0)| 00:00:01 |
|   7 |        NESTED LOOPS                    |                       |      |       |       |            |          |
```

```
|  17 |                 NESTED LOOPS OUTER         |                            |     1 |   539 |       |     5   (0)| 00:00:01 |
|  18 |                  NESTED LOOPS OUTER        |                            |     1 |   340 |       |     5   (0)| 00:00:01 |
|  19 |                   TABLE ACCESS BY INDEX ROWID| PA_STUDENT               |     1 |   316 |       |     3   (0)| 00:00:01 |
|* 20 |                    INDEX UNIQUE SCAN       | PK_STUDENT                 |     1 |       |       |     2   (0)| 00:00:01 |
|  21 |                   TABLE ACCESS BY INDEX ROWID| PA_STUD_USER             |     1 |    24 |       |     2   (0)| 00:00:01 |
|* 22 |                    INDEX UNIQUE SCAN       | PK_STUD_USER               |     1 |       |       |     1   (0)| 00:00:01 |
|  23 |                  TABLE ACCESS BY INDEX ROWID | PA_ORG                   |     1 |   199 |       |     0   (0)|          |
|* 24 |                   INDEX UNIQUE SCAN        | PK_ORG                     |     1 |       |       |     0   (0)|          |
|  25 |                 TABLE ACCESS BY INDEX ROWID  | PA_DOMAIN                |    13 |   533 |       |     1   (0)| 00:00:01 |
|* 26 |                  INDEX UNIQUE SCAN         | PK_DOMAIN                  |     1 |       |       |     0   (0)|          |
|  27 |                TABLE ACCESS BY INDEX ROWID   | PA_USRRF_STUD            |   100 |  3300 |       |     1   (0)| 00:00:01 |
|* 28 |                 INDEX UNIQUE SCAN          | PK_USRRF_STUD              |     1 |       |       |     0   (0)|          |
|  29 |               VIEW PUSHED PREDICATE        | PV_STUD_USER               |     1 |    12 |       |     1   (0)| 00:00:01 |
|* 30 |                FILTER                      |                            |       |       |       |            |          |
|  31 |                 NESTED LOOPS OUTER         |                            |     1 |    22 |       |   264  (11)| 00:00:01 |
|* 32 |                  INDEX UNIQUE SCAN         | PK_STUDENT                 |     1 |    10 |       |     2   (0)| 00:00:01 |
|* 33 |                  MAT_VIEW ACCESS FULL      | PV_AP_STUD_USER            |     1 |    12 |       |   262  (11)| 00:00:01 |
|  34 |              BUFFER SORT                   |                            |     2 |   192 |       |    10   (0)| 00:00:01 |
|  35 |               TABLE ACCESS BY INDEX ROWID  | PA_CPNT_COMPLIANCE_DATA    |     2 |   192 |       |     3   (0)| 00:00:01 |
|* 36 |                INDEX RANGE SCAN            | IX_CPNT_CD__EVTHST         |     2 |       |       |     1   (0)| 00:00:01 |
|  37 |             TABLE ACCESS BY INDEX ROWID    | PA_CPNT_TYPE               |     1 |    61 |       |     1   (0)| 00:00:01 |
|* 38 |              INDEX UNIQUE SCAN             | PK_CPNT_TYPE               |     1 |       |       |     0   (0)|          |
|  39 |            TABLE ACCESS BY INDEX ROWID     | PA_RQMT_TYPE               |     1 |    74 |       |     1   (0)| 00:00:01 |
|* 40 |             INDEX UNIQUE SCAN              | PK_RQMT_TYPE               |     1 |       |       |     0   (0)|          |
```

```
-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                              | Name                  |  Rows  | Bytes |TempSpc| Cost (%CPU)|   Time    |
-----------------------------------------------------------------------------------------------------------------
|  52 |          INDEX FAST FULL SCAN          | IX_STUD_USER__STUDENT |  710K|       |       |  309   (16)| 00:00:01 |
|  53 |        BUFFER SORT                     |                       |  710K|       |       |  264T  (79)|999:59:59 |
|  54 |         INDEX FAST FULL SCAN           | IX_STUD_USER__STUDENT |  710K|       |       |  309   (16)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------------
```

```
|  54 |          INDEX FAST FULL SCAN      | IX_STUD_USER__STUDENT      |  710K|       |       |   309  (16)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------------
```

Who doesn't have a 15 Exabyte Temp Tablespace?
and enough paper in their printer for 156 quadrillion rows?

TidalScale™

Some Times You Need More Resources: The Result Cache Would Help

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

in 60 minutes ...

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?
```

Unfortunately This Is The Most Common Oracle Tuning Methodology

Performance Tuning History (1:3)

- The first thing I learned in IT, writing Fortran on punch cards, was performance
- In 1969 the biggest cost consideration in an application was the cost of cpu
- No different from how we pay today … licensing by cpu core
- And the slowest thing in computing was reading from cards and tape
- Today it is reading from the storage array
- Once we had our data in memory everything was fast
- The more things change the more they stay the same

- Over the decades we have improved networks … from bytes to KB to MB to GB and today we casually talk about 100GB InfiniBand
- We have improved storage not just in TB but in IOPS

Performance Tuning History (2:3)

- Our servers are orders of magnitude faster with the ability to achieve results orders of magnitude faster than in the past
- Still the number one issue we face today is the same: Performance
- Why?

Performance Tuning History (3:3)

- Because …
- Our data sets continue to grow faster than the amount of memory
- Commodity two socket server will get you, at most, 3TB of DRAM and the Oracle Database licenses for that server will cost $1.6M
- Even Exadata Flash is 1/100 the speed of DRAM

- Does anyone have a data warehouse or data lake smaller than 2-3TB?
- Would your management be willing to spend $2M for just one server?

What Doesn’t Work

TidalScale™

What If You Had The Memory You Needed? (1:3)

- Memory management isn't easy: You need is enough memory to provide resources to optimize each of these parameters

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™

What If You Had The Memory You Needed? (2:3)

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

What If You Had The Memory You Needed? (3:3)

Feature/Option	Initialization Parameter	Description
Data Guard	RECV_BUF_SIZE SEND_BUF_SIZE	Buffer at Standby Database receiving redo logs Buffer at Primary Database sending redo logs
In-Memory Database	INMEMORY_EXPRESSIONS_USAGE INMEMORY_FORCE INMEMORY_QUERY INMEMORY_SIZE INMEMORY_VIRTUAL_COLUMNS OPTIMIZER_INMEMORY_AWARE	Controls which IMDB Expressions are populated into the Column Store and are available for queries Force tables to be in-memory Specifies whether in-memory queries are allowed Size in bytes of in-memory area Enable to store virtual columns in the In-Memory area Optimizer in-memory columnar awareness
Real Application Clusters	Linux • RMEM • WMEM Solaris • RSIZE • WSIZE	 Used by TCP to regulate receive buffer sizes Used by TCP to regulate write buffer sizes Used by TCP to regulate receive buffer sizes Used by TCP to regulate write buffer sizes
Result Cache	CLIENT_RESULT_CACHE_SIZE RESULT_CACHE_MAX_RESULT RESULT_CACHE_MAX_SIZE	Specifies the maximum size of the client per-process result set cache Specifies the percentage of RESULT_CACHE_MAX_SIZE that any single result can use Specifies the maximum amount of SGA memory that can be used by the Result Cache

If You Had The Memory You Need?

```
SQL> SELECT COUNT(v1)
  2  FROM t1
  3  WHERE (qty > 495 OR (qty < 3 AND part_no = 50));

----------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)|
----------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |       |       | 14739 (100)|
|   1 |  SORT AGGREGATE     |      |     1 |    19 |            |
|*  2 |   TABLE ACCESS FULL| T1   |  100K|  1862K| 14739    (6)|
----------------------------------------------------------------

SQL> ALTER SYSTEM SET inmemory_size = 500G SCOPE = spfile;
-- restart
SQL> ALTER TABLE t1 INMEMORY PRIORITY high;

-------------------------------------------------------------------------
| Id  | Operation                    | Name | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |      |       |       |  1974 (100)|
|   1 |  SORT AGGREGATE              |      |     1 |    19 |            |
|*  2 |   TABLE ACCESS INMEMORY FULL| T1   |  100K|  1862K|  1974   (44)|
-------------------------------------------------------------------------
Source: Jonathan Lewis, Oracle Scratchpad, Oct. 20, 2016
```

- Only 500GB more and you could cut the cost by 90%

If You Had The Memory You Need?

```
SQL> sho parameter private

NAME                          TYPE    VALUE
----------------------------- ------- ---------
private_temp_table_prefix     string  ORA$PTT_


SQL> CREATE PRIVATE TEMPORARY TABLE ora$ptt_ocdr(
  2  rid    NUMBER(10),
  3  rname VARCHAR2(20))
  4  ON COMMIT PRESERVE DEFINITION;

SQL> SELECT table_name, tablespace_name
  2  FROM dba_private_temp_tables

TABLE_NA.     TABLESPACE_NAME
------------- --------------
ORA$PTT_OCDR  TEMP
```

Source: Morgan's Library, Private Temporary Tables, Mar. 14, 2018

What Does Not Work: Buying Huge Servers

- To purchase servers with large memory footprints requires buying servers with lots of sockets and filling them with lots of cpu cores
- The more cores the more memory ... the more licenses
- And the higher the annual support cost
- To get large amounts of memory requires purchases that can only be afforded by the largest organizations and for only their most mission critical needs
- The largest amount of memory you can get with a commodity 2 socket server like a Dell R640 is 3TB and that will get you 16 cores and a list price, just for Oracle licenses, of $380,000 ... Exceed 3TB DRAM and you don't get a quantity discount
- For high availability by 2 because you need 2 nodes for RAC
- Multiply to 2 again because you need to replicate your cluster for DR
- That's a lot of hardware, a lot of money, you have 224 cores, 112 licenses, 12TB of memory and your largest possible SGA is smaller than 3TB

TidalScale™

What Does Not Work: Moving To An In-Memory Database

- SAP HANA
  - Requires that you recreate DDL
  - Requires that you rewrite every SQL statement
  - Requires that you throw away your PL/SQL
  - Requires you to reinvent your deployment, patching, and monitoring processes and procedures
  - Requires that you throw away your entire investment in Oracle perpetual licenses
- Oracle TimesTen
  - Requires that you rearchitect your schema to make it hierarchical
  - Requires that you reinvent your deployment, patching, and monitoring processes and procedures
  - Requires that you throw away a sizable percentage of your investment in Oracle licenses
- And not just databases
  - Any workload running on essentially any flavor of Linux
    - R and Business Analytics
    - GoldenGate and Data Integration

In-Memory Databases

- We all know that Oracle has solutions for our performance
- There isn't an Oracle DBA that isn't aware of legacy features such as the Buffer Cache, Java Cache, PGA, Result Cache, Sort Area, and Streams Cache … that can be tuned to improve performance
- Most DBAs know that with 12c Oracle introduced Full Database Caching, Database In-Memory Database (DBIM) and In-Memory Aggregation … why aren't we using these features?
- And many in this room are aware of new 18c features such as the Memoptimize Pool and Private Temporary Tables
- What is stopping us from using these and ending our performance issues?
- Lack of memory
- Because to get the memory we would require is prohibitively expensive

The Software Defined Server Solution

TidalScale™

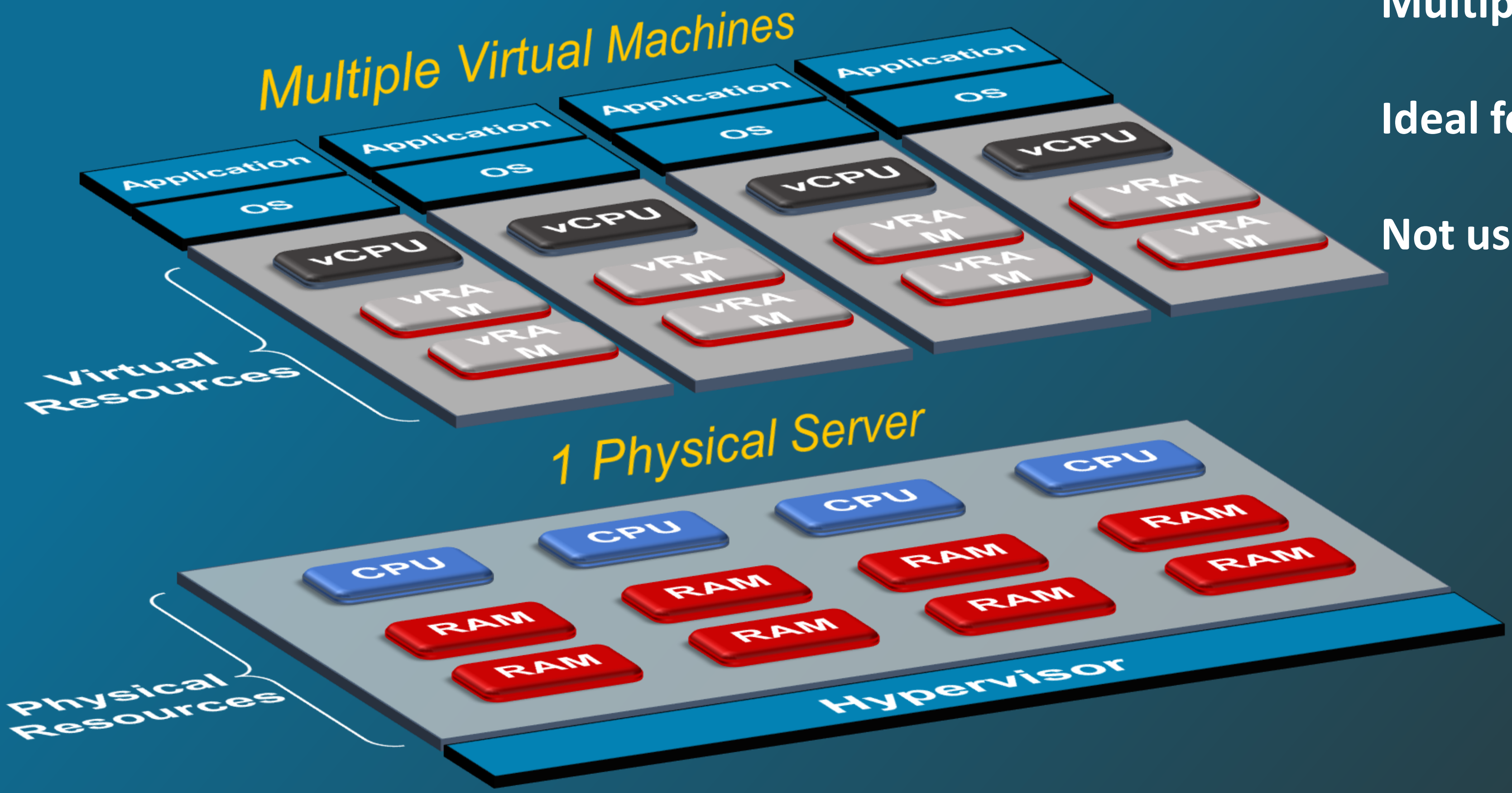
What Works

- TidalScale's founders realized that the only way to finally address the issue was to create a disruptive technology
- One that would meet the needs of Operations for stability and performance
- One that would meet the needs of Development for agility and flexibility
- One that would be based on the DevOps principle of Software Defined Everything
- One that would be affordable for Oracle's customers

What TidalScale Is

- With TidalScale's HyperKernel hard partitioning software, organizations can combine one or more commodity servers into a single Software-Defined Server
- Use all the aggregated resources of the combined servers – memory, CPUs, I/O and networks – as a single system logical server – and at the same time use hard partitioning to limit cpu cores
- TidalScale solutions deliver in-memory performance at any scale, self-optimize through machine learning, and use standard hardware
- Compatible with all applications running on Linux without modification
- You can create a software defined server across multiple bare metal servers
  - With just 3 mouse clicks
- Aggregate memory up to 64TB
- Boosts throughput by more than 20X
- TidalScale hard partitioning gives you the ability to control cpu core licensing by licensing what you need, when you need it, not every core in the server

Legacy Virtualization



Discrete physical resources

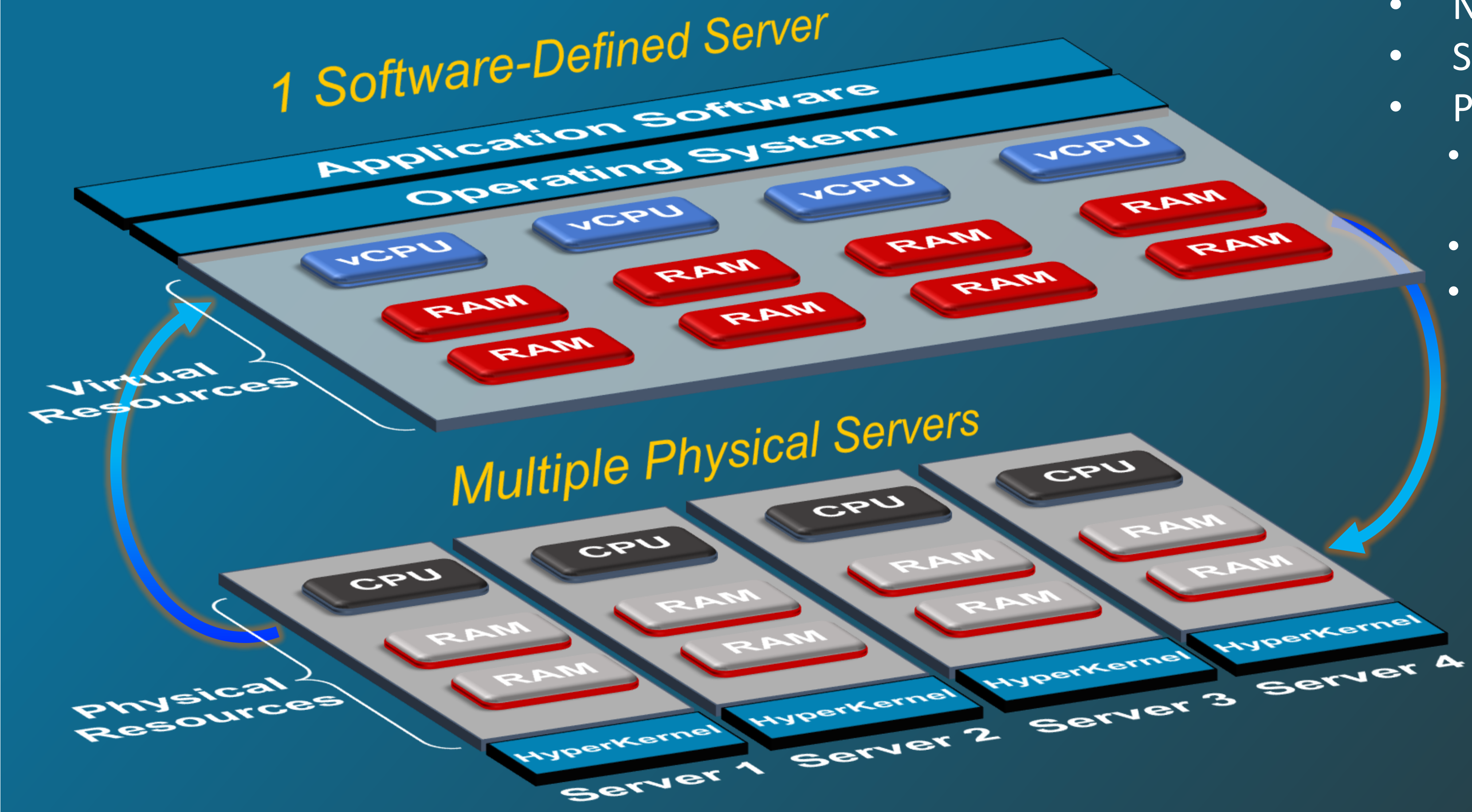
Multiple Guest Operating Systems

Ideal for small applications and data

Not useful for large-scale applications

TidalScale

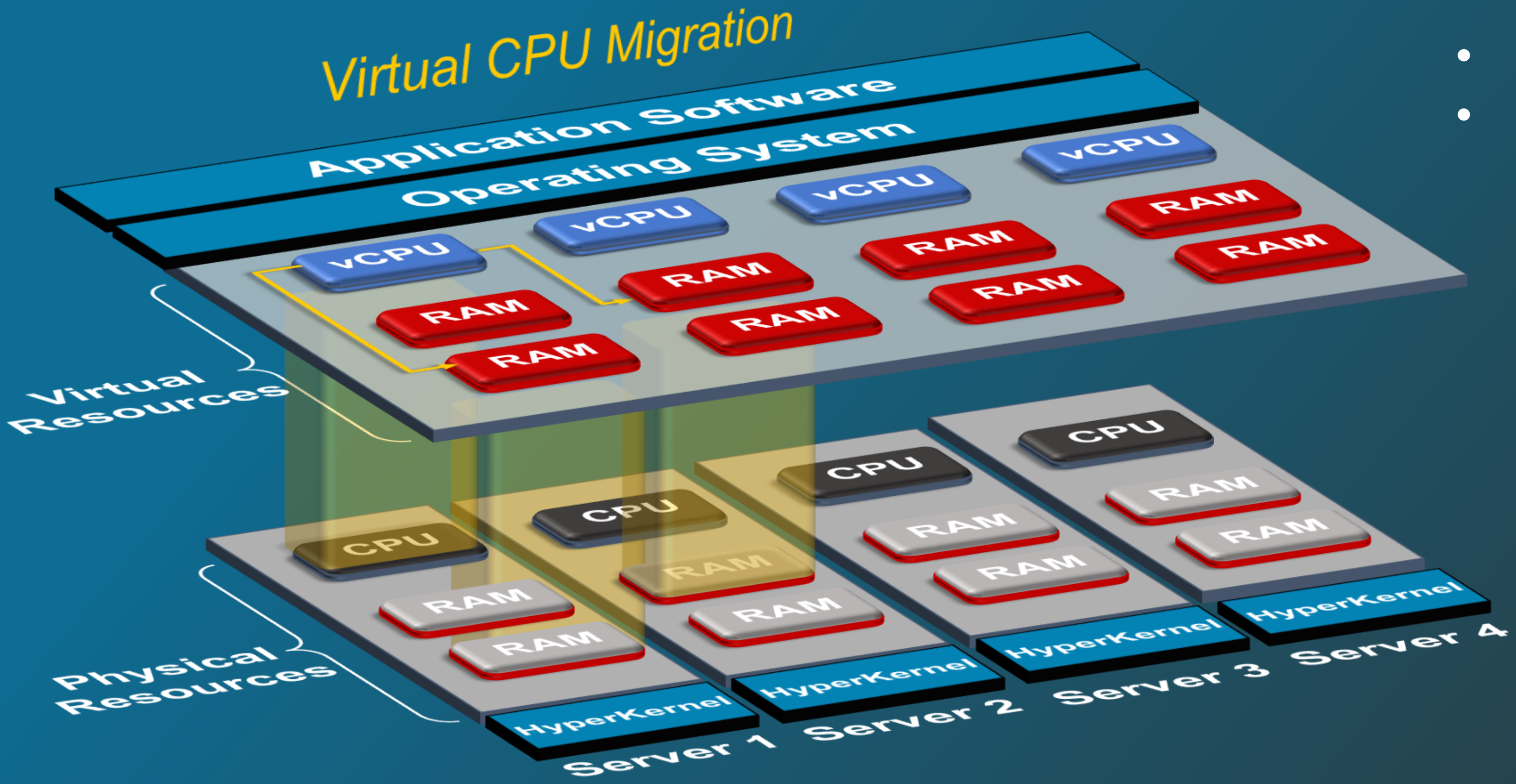
Software Defined Servers



- Aggregated physical resources
- Single guest operating system
- No O/S or application modifications
- Support for Enterprise Size Workloads
- Patented Machine Learning Algorithm
  - Maps virtual resources, 1:1, to physical infrastructure
  - Dynamic Runtime Load balancing
  - Transparent vCPU and memory page migration

TidalScale

Machine Learning Driven Self Optimization

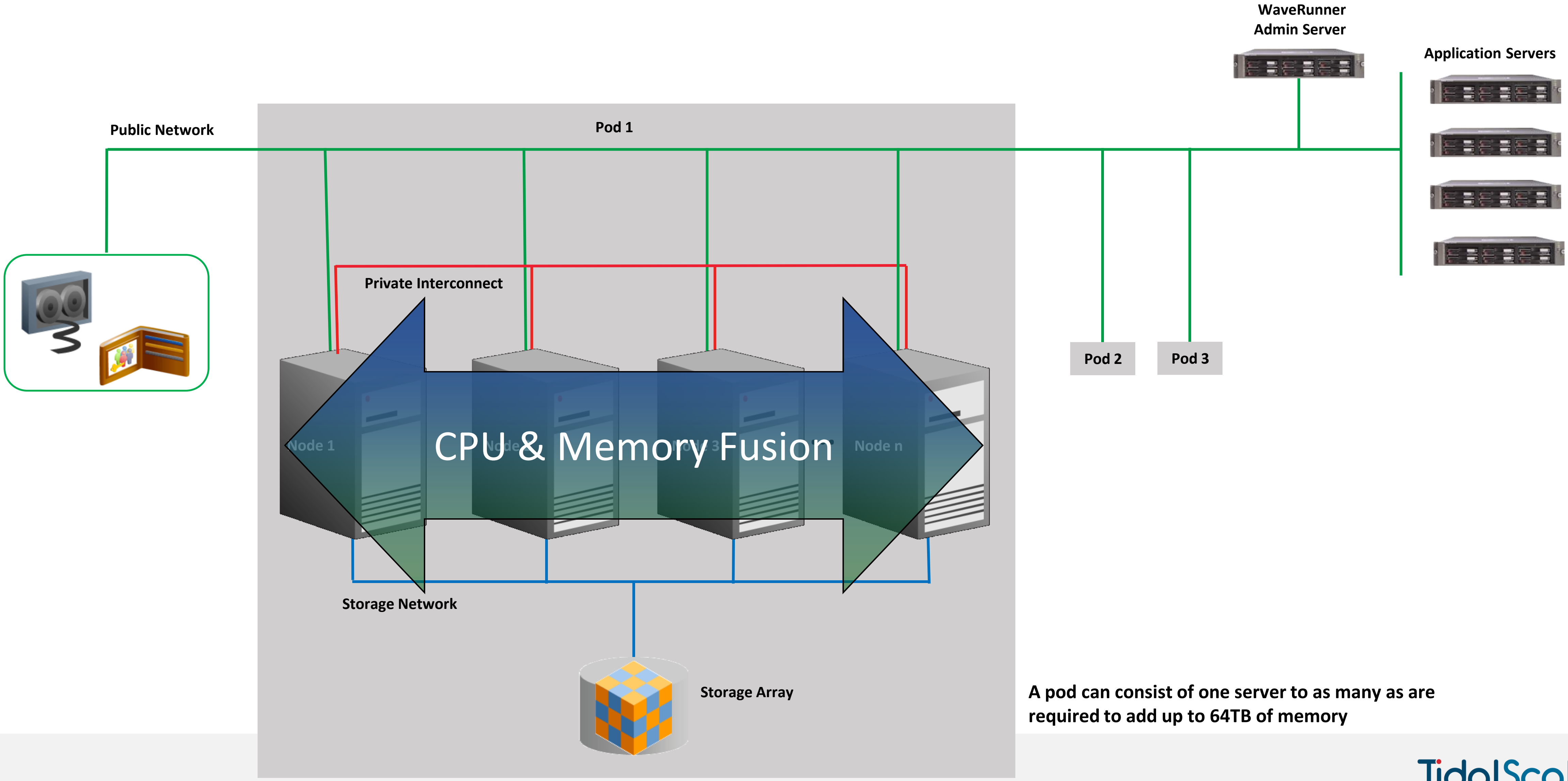


Patented Machine Learning Algorithm:

- Maps virtual resources, 1:1, to physical infrastructure
- Dynamic Runtime Load balancing
- Transparent vCPU and memory page migration

TidalScale

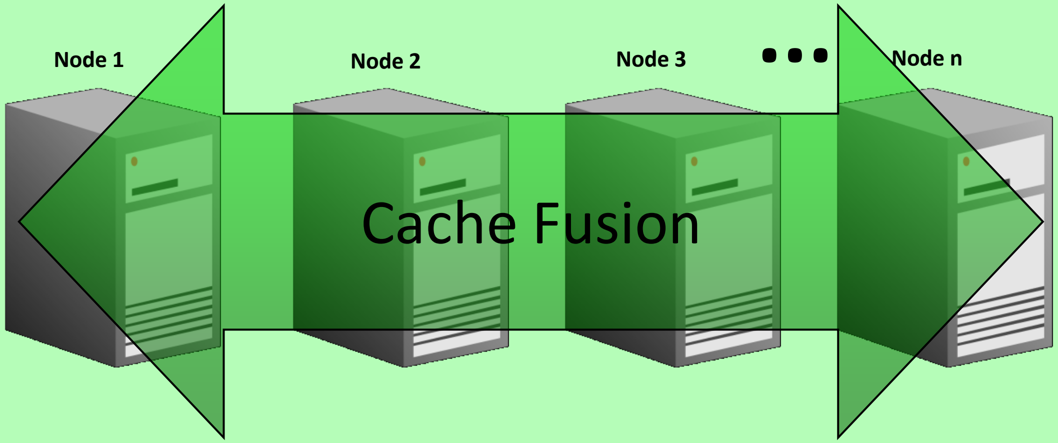
TidalScale Architecture



A pod can consist of one server to as many as are required to add up to 64TB of memory

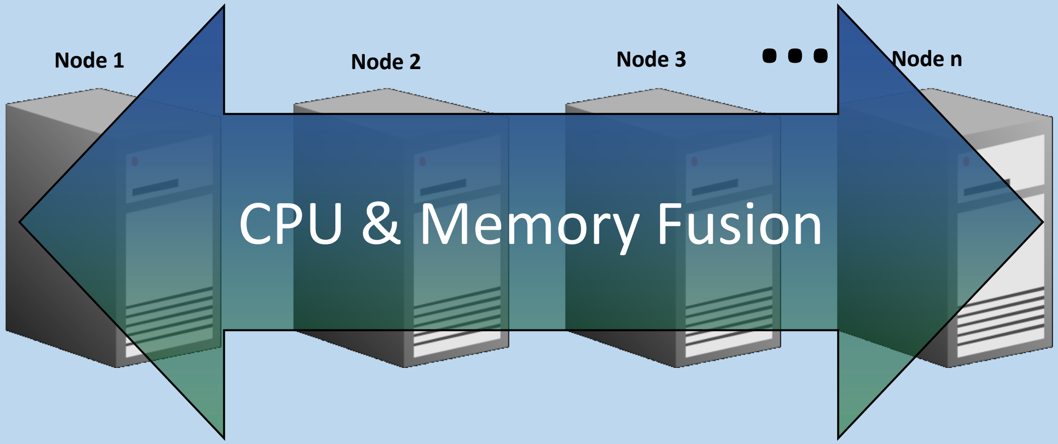
TidalScale™

Oracle RAC vs TidalScale



ORACLE RAC

License and Support for every cpu core on every physical server



TidalScale™

License and Support only the cpu cores you need

This Is What 1TB Of Memory Looks Like To Most DBAs and CFOs

WORKLOAD REPOSITORY COMPARE PERIOD REPORT

Snapshot Set	DB Name	DB Id	Instance	Inst num	Release	Cluster	Host	Std Block Size
First (1st)	ORAP09	2886124853	orap09	1	11.2.0.3.0	NO	db20p03sh	8192
Second (2nd)	ORAP09	2886124853	orap09	1	11.2.0.3.0	NO	db20p03sh	8192

Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Avg Active Users	Elapsed Time (min)	DB time (min)
1st	33759	02-Apr-14 04:00:19 (Wed)	33760	02-Apr-14 04:30:22 (Wed)	0.8	30.1	24.5
2nd	33807	03-Apr-14 04:00:13 (Thu)	33808	03-Apr-14 04:30:16 (Thu)	1.0	30.1	29.8
%Diff					*22.2*	*0.0*	*21.6*

Host Configuration Comparison

	1st	2nd	Diff	%Diff
Number of CPUs:	80	80	0	0.0
Number of CPU Cores:	40	40	0	0.0
Number of CPU Sockets:	4	4	0	0.0
Physical Memory:	1031464.9M	1031464.9M	0M	0.0
Load at Start Snapshot:	19.09	20.68	1.59	8.3
Load at End Snapshot:	11.49	11.18	-.31	-2.7
%User Time:	14.88	14.44	-.44	-3.0
%System Time:	1.08	1.05	-.03	-2.8
%Idle Time:	83.92	84.38	.46	0.5
%IO Wait Time:	.31	.45	.13	45.2

- 4 Sockets
- 40 cpu cores
- 80 threads
- 20 Oracle EE licenses ($950,000)
- 20 Diag & Tuning Licenses ($250,000)
- Total Licenses: $1,200,000
- Annual Support: $264,000

And all you get is 1TB of memory

TidalScale™

TidalScale Metrics

- We used HammerDB to build three identical 500GB Oracle 12.2.0.1 databases on Oracle Enterprise Linux 7.4
  - Env 1: 512GB RAM … Oracle installation performed by OUI and DBCA <next><next><next>
  - Env 2: 1024GB RAM ... A 2 node TidalScale pod with Database In-Memory enabled
- Nothing was customized
- The databases are 500GB and identical ... created with the same script
- Adaptive Queries were not disabled … but we've tested both ways
- Evolving Baselines were not disabled … but we've tested both ways
- No Explain Plans were run
- No SQL Tuning was performed
- Every DML statement was generic TPC-H benchmark
- Are you ready to view the results?

Test Results: Timing

Env 1: Run time – 13,249 seconds

Env 2: Run time – 558 seconds

A 25X performance improvement

Test Results: AWR Report (1:4)

Env 1: Bare Metal

DB Name	DB Id	Unique Name	Role	Edition	Release	RAC	CDB
ORCL	1512797244	orcl	PRIMARY	EE	12.2.0.1.0	NO	NO

Instance	Inst Num	Startup Time
oracle	1	24-Aug-18 15:08

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
oracle201	Linux x86 64-bit	32	16	2	503.81

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	42	26-Aug-18 05:29:27	82	1.0
End Snap:	43	26-Aug-18 05:41:24	78	1.0

Env 2: TidalScale 2 Node Pod

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)		

Test Results: AWR Report (2:4)

Env 1: Bare Metal

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	12.4	442.6	2.14	0.32
DB CPU(s):	1.5	53.5	0.26	0.04
Background CPU(s):	0.0	0.3	0.00	0.00
Redo size (bytes):	7,014.7	251,425 .4		
Logical read (blocks):	80,812.5	2,896,54 9.7		
Block changes:	41.2	1,476.5		
Physical read (blocks):	78,627.8	2,818,2 43.6		
Physical write (blocks):	3,969.4	142,275.2		
Read IO requests:	1,012.3	36,284.8		
Write IO requests:	141.6	5,074.4		
Read IO (MB):	614.3	22,017.5		
Write IO (MB):	31.0	1,111.5		
IM scan rows:	0.0	0.0		
Session Logical Read IM:	0.0	0.0		
User calls:	38.1	1,366.7		
Parses (SQL):	4.9	174.2		
Hard parses (SQL):	0.4	13.3		
SQL Work Area (MB):	10.2	364.0		
Logons:	0.2	6.1		
Executes (SQL):	5.8	206.8		
Rollbacks:	0.0	0.0		
Transactions:	0.0			

Env 2: TidalScale 2 Node Pod

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	11.4	202.9	1.83	0.01
DB CPU(s):	11.2	199.8	1.80	0.01
Background CPU(s):	0.0	0.5	0.00	0.00
Redo size (bytes):	5,896.7	105,093.1		
Logical read (blocks):	1,673,100.2	29,818,686.1		
Block changes:	32.4	576.7		
Physical read (blocks):	644.7	11,489.2		
Physical write (blocks):	1,069.5	19,061.1		
Read IO requests:	21.0	374.4		
Write IO requests:	35.2	627.2		
Read IO (MB):	5.0	89.8		
Write IO (MB):	8.4	148.9		
IM scan rows:	49,445,926.5	881,245,804.0		
Session Logical Read IM:	1,669,664.9	29,757,459.9		
User calls:	976.6	17,405.8		
Parses (SQL):	4.5	80.2		
Hard parses (SQL):	0.3	4.5		
SQL Work Area (MB):	82.8	1,476.3		
Logons:	1.0	18.1		
Executes (SQL):	6.2	111.1		
Rollbacks:	0.0	0.0		
Transactions:	0.1			

- When you have sufficient memory logical reads replace far slower physical reads

Test Results: AWR Report (3:4)

Env 1: Bare Metal

Memory Statistics

	Begin	End
Host Mem (MB):	515,896.9	515,896.9
SGA use (MB):	155,136.0	155,136.0
PGA use (MB):	4,019.2	15,494.7
% Host Mem used for SGA+PGA:	30.85	33.07

Env 2: TidalScale 2 Node Pod

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

Test Results: AWR Report (4:5)

Env 1: Bare Metal

Cache Sizes

	Begin	End		
Buffer Cache:	131,584M	131,584M	Std Block Size:	8K
Shared Pool Size:	15,251M	15,249M	Log Buffer:	495,120K
In-Memory Area:	0M	0M		

Env 2: TidalScale 2 Node Pod

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		

- With TidalScale there is sufficient memory to put 500GB of segments into memory

Test Results: AWR Report (5:5)

Segments by Physical Reads

Env 1: Bare Metal

- Total Physical Reads: 56,364,871
- Captured Segments account for 99.0% of Total
- When ** MISSING ** occurs, some of the object attributes may not be available

Owner	Tablespace Name	Object Na me	Subobject Name	Obj. Type	Obj#	Dataobj#	Physical Reads	%Total
TPCH	TPCH500	LINEITEM		TABLE	73611	73611	39,610,254	70.27
TPCH	TPCH500	ORDERS		TABLE	73604	73604	11,567,078	20.52
TPCH	TPCH500	PART		TABLE	73607	73607	4,361,323	7.74
TPCH	TPCH500	PARTSUPP		TABLE	73605	73605	272,313	0.48
SYS	SYSAUX	WRH$_ACTIVE_SESSION_HISTORY	WRH$_ACTIVE_SESSION_HISTORY_1512797244_24	TABLE PARTITION	74028	74028	32	0.00

Segments by Physical Reads

Env 2: TidalScale 2 Node Pod

- Total Physical Reads: 471,058
- Captured Segments account for 0.0% of Total
- When ** MISSING ** occurs, some of the object attributes may not be available

Owner	Tablespace Name	Object Na me	Subobject Name	Obj. Type	Obj#	Dataobj#	Physical Reads	%Total
SYS	SYSAUX	WRH$_ACTIVE_SESSION_HISTORY	WRH$_ACTIVE_SESSION_HISTORY_1499046141_2706	TABLE PARTITION	98557	98557	113	0.02
SYS	SYSAUX	SYS_LOB0000010641C00038$$		LOB	10642	10642	32	0.01
SYS	SYSAUX	WRH$_SQL_PLAN_PK		INDEX	10644	10644	11	0.00
SYS	SYSAUX	WRH$_ENQUEUE_STAT_PK		INDEX	10676	10676	2	0.00
SYS	SYSAUX	WRH$_SYSMETRIC_SUMMARY		TABLE	10843	10843	2	0.00

- TPCH tables are in memory so physical reads from TPCH500 are negligible

Wrap Up

TidalScale™

Wrap Up

- Memory is faster than disk
- Memory is faster than flash
- A larger percentage of your workloads require memory that exceeds what is available on commodity servers
- If you can put your critical data into memory performance will improve dramatically and faster than data sets grow
- If you are ready to put an end to performance issues in your DW, DSS and DL systems you now know how to do so
- If you want to stop working 60+ hours a week this may help
- TidalScale Software Defined Servers are the only solution for getting the memory you need at a price your organization can afford

Thank you

daniel.morgan@tidalscale.com / 206-669-2949

Email me for a copy of the init.ora, DDL, DML, AWR Reports, and this presentation

TidalScale™