Oracle中的AWR，全称为Automatic Workload Repository，自动负载信息库。它收集关于特定数据库的操作统计信息和其他统计信息，Oracle以固定的时间间隔(默认为1个小时)为其所有重要的统计信息和负载信息执行一次快照，并将快照存放入AWR中。这些信息在AWR中保留指定的时间(默认为1周)，然后执行删除。执行快照的频率和保持时间都是可以自定义的。

Database Information

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
RACDB	907945437	racdb1	1	10.2.0.1.0	YES	rac1

DB Time

表示用户操作花费的时间，包括CPU时间和等待事件。通常同时这个数值判读数据库的负载情况。

db time = cpu time + wait time(不包含空闲等待)(非后台进程)

*db time就是记录的服务器花在数据库运算(非后台进程)和等待(非空闲等待)上的时间。对应于V$SESSION的elapsed_time字段累积。

```
--查看最近7天的DB Time
 WITH sysstat AS
  (select sn.begin_interval_time begin_interval_time,
          sn.end_interval_time end_interval_time,
          ss.stat_name stat_name,
          ss.value e_value,
          lag(ss.value, 1) over(order by ss.snap_id) b_value
     from dba_hist_sysstat ss, dba_hist_snapshot sn
    where trunc(sn.begin_interval_time) >= sysdate - 7
      and ss.snap_id = sn.snap_id
      and ss.dbid = sn.dbid
      and ss.instance_number = sn.instance_number
      and ss.dbid = (select dbid from v$database)
      and ss.instance_number = (select instance_number from
v$instance)
      and ss.stat_name = 'DB time')
 select to_char(BEGIN_INTERVAL_TIME, 'mm-dd hh24:mi') ||
        to_char(END_INTERVAL_TIME, 'hh24:mi') date_time,
        stat_name,
        round((e_value - nvl(b_value, 0)) /
```

```
20              (extract(day from(end_interval_time -
    begin_interval_time)) * 24 * 60 * 60 +
21              extract(hour from(end_interval_time -
    begin_interval_time)) * 60 * 60 +
22              extract(minute from(end_interval_time -
    begin_interval_time)) * 60 +
23              extract(second from(end_interval_time -
    begin_interval_time))),
24              0) per_sec
25    from sysstat
26   where (e_value - nvl(b_value, 0)) > 0
27     and nvl(b_value, 0) > 0;
```

Snapshot Information

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	222	11-Sep-17 06:55:24	29	2.0
End Snap:	223	11-Sep-17 08:00:26	32	2.1
Elapsed:		65.04 (mins)		
DB Time:		0.13 (mins)		

(1)Sessions
表示采集实例连接的会话数。这个数可以帮助我们了解数据库的并发用户数大概的情况。
(2)Cursors/session
每个会话平均打开的游标数。
(3)Elapsed
通过Elapsed/DB Time比较，反映出数据库的繁忙程度。如果DB Time>>Elapsed，则说明数据库很忙。

注：需要注意的是AWR是一个数据合集。比如在1分钟之内，1个用户等待了30秒钟，那么10个用户等待事件就是300秒。CPU时间也是一样，在1分钟之内，1个CPU处理30秒钟，那么4个CPU就是120秒。这些时间都是以累积的方式记录在AWR当中的。
示例
DB CPU——这是一个用于衡量CPU的使用率的重要指标。假设系统有N个CPU，那么如果CPU全忙的话，一秒钟内的DB CPU就是N秒。除了利用CPU进行计算外，数据库还会利用其它计算资源，如网络、硬盘、内存等等，这些对资源的利用同样可以利用时间进行度量。假设系统有M个session在运行，同一时刻有的session可能在利用CPU，有的session可能在访问硬盘，那么在一秒钟内，所有session的时间加起来

就可以表征系统在这一秒内的繁忙程度。一般的，这个和的最大值应该为M。这其实就是Oracle提供的另一个重要指标:DB time，它用以衡量前端进程所消耗的总时间。

对除CPU以外的计算资源的访问，Oracle用等待事件进行描述。同样地，和CPU可分为前台消耗CPU和后台消耗CPU一样，等待事件也可以分为前台等待事件和后台等待事件。DB Time一般的应该等于"DB CPU + 前台等待事件所消耗时间"的总和。等待时间通过v$system_event视图进行统计，DB Time和DB CPU则是通过同一个视图，即v$sys_time_model进行统计。

"Load Profile"中关于DB Time的描述

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	11.7	55.2	0.22	1.31
DB CPU(s):	7.1	33.4	0.13	0.79

假如此处CPU个数为8，可以知道前台进程用系统CPU的比例为7.1/8=88.75%，7.1为DB CPU(s)/Per Second，DB time/s为11.7，可以看出这个系统是CPU非常繁忙的。里面CPU占了7.1，则其它前台等待事件占了11.7 – 7.1 = 4.6 Wait Time/s。DB Time占 DB CPU的比重: 7.1/11.7= 60.68%

"Top 5 Timed Events"中关于DB CPU的描述

按照CPU/等待事件占DB Time的比例大小，这里列出了Top 5。如果一个工作负载是CPU繁忙型的话，那么在这里应该可以看到 DB CPU的影子。

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		6,475		60.45	
external table read	129,744	647	5	6.04	User I/O
direct path write	231,625	272	1	2.54	User I/O
PX Deq: reap credit	34,274	63	2	0.59	Other
PX Deq: Slave Session Stats	1,596	43	27	0.40	Other

我们刚刚已经算出了DB CPU 的%DB time，60%。其它的external table read、direct path write、PX Deq: read credit、PX Deq: Slave Session Stats这些就是占比重40的等待事件里的Top 4了。

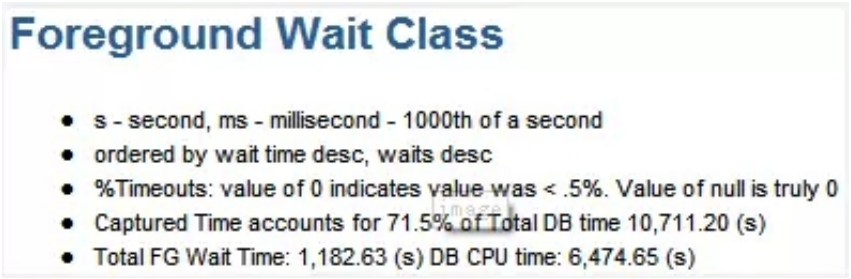
"Top 5 Timed Foreground Events"的局限性

再研究下这个Top 5 Timed Foreground Events，如果先不看Load Profile，是不能计

算出一个CPU-Bound的工作负载。要知道系统CPU的繁忙程序，还要知道这个AWR所基于两个snapshot的时间间隔，还要知道系统CPU的个数。记住CPU利用率 = DB CPU/(CPU_COUNT*Elapsed TIME)。这个Top 5 给我们的信息只是这个工作负载应该是并行查询，从外部表读取数据，并用insert append的方式写入磁盘，同时，主要时间耗费在CPU的运算上。

"DB Time" > "DB CPU" + "前台等待事件所消耗时间" ——进程排队时间

上面提到，DB Time一般的应该等于DB CPU + 前台等待事件所消耗时间的总和。在下面有对这三个值的统计：



DB CPU = 6474.65

DB TIME = 10711.2

FG Wait Time = 1182.63

明显的，DB CPU + FG Wait Time < DB Time，只占了71.5%

其它的28.5%被消耗到哪里去了呢?这里隐含着一个Oracle如何计算DB CPU和DB Time的问题。当CPU很忙时，如果系统里存在着很多进程，就会发生进程排队等待CPU的现象。在这样，DB TIME是把进程排队等待CPU的时间算在内的，而DB CPU是不包括这一部分时间。这是造成 DB CPU + FG Wait Time < DB Time的一个重要原因。如果一个系统CPU不忙，这两者应该就比较接近。在这个例子中，这是一个CPU非常繁忙的系统，而71.5%就是一个信号，它提示着这个系统可能是一个CPU瓶颈的系统。

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	7,584M	7,584M	Std Block Size:	8K
Shared Pool Size:	2,336M	2,336M	Log Buffer:	14,340K

这部分列出AWR在性能采集开始和结束的时候，数据缓冲池(buffer cache)和共享池(shared pool)的大小。通过对比前后的变化，可以了解系统内存消耗的变化。数据取

自dba_hist_parameter。

Load Profile

	Per Second	Per Transaction
Redo size:	140,233.44	105,126.86
Logical reads:	7,052.13	5,286.67
Block changes:	300.78	225.48
Physical reads:	590.22	442.46
Physical writes:	14.74	11.05
User calls:	655.05	491.06
Parses:	182.11	136.52
Hard parses:	17.60	13.20
Sorts:	9.48	7.10
Logons:	0.64	0.48
Executes:	278.48	208.77
Transactions:	1.33	

% Blocks changed per Read:	4.27	Recursive Call %:	12.72
Rollback per transaction %:	76.02	Rows per Sort:	7.98

这两部分是数据库资源负载的一个明细列表，分隔成每秒钟的资源负载和每个事务的资源负载。

Redo size

每秒(每个事务)产生的日志大小(单位字节)

Logical reads

每秒(每个事务)产生的逻辑读(单位是block)。在很多系统里select执行次数要远远大于transaction次数。这种情况下，可以参考Logical reads/Executes。在良好的oltp环境下，这个应该不会超过50，一般只有10左右。如果这个值很大，说明有些语句需要优化。

Block Changes

每秒(每个事务)改变的数据块数。

Physical reads

每秒(每个事务)产生的物理读(单位是block)。一般物理读都会伴随逻辑读，除非直接读取这种方式，不经过cache。

Physical writes

每秒(每个事务)产生的物理写(单位是block)。

User calls

每秒(每个事务)用户调用次数。User calls/Executes基本上代表了每个语句的请求次数，Executes越接近User calls越好。

Parses

每秒(每个事务)产生的解析(或分析)的次数，包括软解析和硬解析，但是不包括快速软解析。软解析每秒超过300次意味着你的"应用程序"效率不高，没有使用soft soft parse，调整session_cursor_cache。

Hard parses

每秒(每个事务)产生的硬解析次数。每秒超过100次，就可能说明你绑定使用的不好。

Sorts

每秒(每个事务)排序次数。

Logons

每秒(每个事务)登录数据库次数。

Executes

每秒(每个事务)SQL语句执行次数。包括了用户执行的SQL语句与系统执行的SQL语句，表示一个系统SQL语句的繁忙程度。

Transactions

每秒的事务数。表示一个系统的事务繁忙程度。目前已知的最繁忙的系统为淘宝的在线交易系统，这个值达到了1000。

% Blocks changed per Read

表示逻辑读用于只读而不是修改的块的比例。如果有很多PLSQL，那么就会比较高。

Rollback per transaction %

看回滚率是不是很高，因为回滚很耗资源。

Recursive Call %

递归调用SQL的比例，在PL/SQL上执行的SQL称为递归的SQL。

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.87	Redo NoWait %:	100.00
Buffer Hit %:	91.63	In-memory Sort %:	100.00
Library Hit %:	86.61	Soft Parse %:	90.33
Execute to Parse %:	34.61	Latch Hit %:	99.94
Parse CPU to Parse Elapsd %:	15.09	% Non-Parse CPU:	94.99

这个部分是内存效率的统计信息。对于OLTP系统而言，这些值都应该尽可能地接近100%。对于OLAP系统而言，意义不太大。在OLAP系统中，大查询的速度才是对性能影响的最大因素。

Buffer Nowait %

非等待方式获取数据块的百分比。
这个值偏小，说明发生SQL访问数据块时数据块正在被别的会话读入内存，需要等待这个操作完成。发生这样的事情通常就是某些数据块变成了热块。
Buffer Nowait<99%说明，有可能是有热块(查找xbh的tch和$v$latch_children的cache buffers chains)。

Redo NoWait %

非等待方式获取redo数据百分比。

Buffer Hit %

数据缓冲命中率，表示了数据块在数据缓冲区中的命中率。
Buffer Hit<95%，可能是要加db_cache_size，但是大量的非选择的索引也会造成该值很高(大量的db file sequential read)。

In-memory Sort %

数据块在内存中排序的百分比。总排序中包括内存排序和磁盘排序。当内存中排序空间不足时，使用临时表空间进行排序，这个是内存排序对总排序的百分比。
过低说明有大量排序在临时表空间进行。在oltp环境下，最好是100%。如果太小，可以调整PGA参数。

Library Hit %

共享池中SQL解析的命中率。
Library Hit<95%，要考虑加大共享池，绑定变量，修改cursor_sharing等。

Soft Parse %

软解析占总解析数的百分比。可以近似当作sql在共享区的命中率。
这个数值偏低，说明系统中有些SQL没有重用，最优可能的原因就是没有使用绑定变量。
<95%：需要考虑到绑定
<80%：那么就可能sql基本没有被重用

Execute to Parse %

执行次数对分析次数的百分比。
如果该值偏小，说明解析(硬解析和软解析)的比例过大，快速软解析比例小。根据实际情况，可以适当调整参数session_cursor_cache，以提高会话中sql执行的命中率。
round(100*(1-:prse/:exe),2) 即(Execute次数 - Parse次数)/Execute次数 x 100%
prse = select value from v$sysstat where name = 'parse count (total)';
exe = select value from v$sysstat where name = 'execute count';

没绑定导致不能重用是一个原因，也有可能sharedpool太小，单纯的加session_cached_cursors也不是根治的办法，不同的sql还是不能重用，还要解析。即使是soft parse 也会被统计入 parse count,所以这个指标并不能反应出fast soft(pga中)/soft (shared pool中)/hard (shared pool 中新解析) 几种解析的比例。只有在pl/sql的类似循环这种程序中使用使用变量才能避免大量parse，所以这个指标跟是否使用bind并没有必然联系增加session_cached_cursors 是为了在大量parse的情况下把soft转化为fast soft而节约资源。

Latch Hit %

latch的命中率。
其值低是因为shared_pool_size过大或没有使用绑定变量导致硬解析过多。要确保>99%，否则存在严重的性能问题，比如绑定等会影响该参数。

Parse CPU to Parse Elapsd %

解析总时间中消耗CPU的时间百分比。即:100*(parse time cpu / parse time elapsed)
解析实际运行事件/(解析实际运行时间+解析中等待资源时间)，越高越好。

% Non-Parse CPU

CPU非分析时间在整个CPU时间的百分比。
100*(parse time cpu / parse time elapsed)= Parse CPU to Parse Elapsd %
查询实际运行时间/(查询实际运行时间+sql解析时间)，太低表示解析消耗时间过多。

Shared Pool Statistics

	Begin	End
Memory Usage %:	41.40	41.87
% SQL with executions>1:	58.32	71.89
% Memory for SQL w/exec>1:	59.42	64.02

Memory Usage %

共享池内存使用率。
应该稳定在70%-90%间，太小浪费内存，太大则内存不足。

% SQL with executions>1

执行次数大于1的SQL比率。
若太小可能是没有使用绑定变量。

% Memory for SQL w/exec>1

执行次数大于1的SQL消耗内存/所有SQL消耗的内存(即memory for sql with execution > 1)。

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
db file sequential read	6,944,214	48,669	7	42.2	User I/O
CPU time		26,469		22.9	
db file scattered read	2,656,393	15,079	6	13.1	User I/O
gc cr block busy	2,453,031	11,332	5	9.8	Cluster
gc buffer busy	782,827	5,141	7	4.5	Cluster

```
--计算topN Timed Foreground Events
SELECT EVENT,
        WAITS,
        round(TIME) "Time(s)",
        round(DECODE(WAITS,
                     NULL,
                     TO_NUMBER(NULL),
                     0,
                     TO_NUMBER(NULL),
                     TIME / WAITS * 1000)) "Avg Wait(ms)",
        round(PCTWTT, 1) "% Total Call Time",
        WAIT_CLASS "Wait Class"
   FROM (SELECT EVENT, WAITS, TIME, PCTWTT, WAIT_CLASS
```

```
14            FROM (SELECT E.EVENT_NAME EVENT,
15                         E.TOTAL_WAITS - NVL(B.TOTAL_WAITS, 0) WAITS,
16                         (E.TIME_WAITED_MICRO -
NVL(B.TIME_WAITED_MICRO, 0)) /
17                         1000000 TIME,
18                         100 *
19                         (E.TIME_WAITED_MICRO -
NVL(B.TIME_WAITED_MICRO, 0)) /
20                         ((SELECT sum(value)
21                             FROM DBA_HIST_SYS_TIME_MODEL e
22                            WHERE e.SNAP_ID = &end_snap
23                              AND e.DBID = &DBID
24                              AND e.INSTANCE_NUMBER = &INST_NUM
25                              AND e.STAT_NAME = 'DB time') -
26                         (SELECT sum(value)
27                             FROM DBA_HIST_SYS_TIME_MODEL b
28                            WHERE b.SNAP_ID = &beg_snap
29                              AND b.DBID = &DBID
30                              AND b.INSTANCE_NUMBER = &INST_NUM
31                              AND b.STAT_NAME = 'DB time')) PCTWTT,
32                         E.WAIT_CLASS WAIT_CLASS
33                    FROM DBA_HIST_SYSTEM_EVENT B,
DBA_HIST_SYSTEM_EVENT E
34                   WHERE B.SNAP_ID(+) = &beg_snap
35                     AND E.SNAP_ID = &end_snap
36                     AND B.DBID(+) = &DBID
37                     AND E.DBID = &DBID
38                     AND B.INSTANCE_NUMBER(+) = &INST_NUM
39                     AND E.INSTANCE_NUMBER = &INST_NUM
40                     AND B.EVENT_ID(+) = E.EVENT_ID
41                     AND E.TOTAL_WAITS > NVL(B.TOTAL_WAITS, 0)
42                     AND E.WAIT_CLASS != 'Idle'
43                  UNION ALL
44                  SELECT 'CPU time' EVENT,
45                         TO_NUMBER(NULL) WAITS,
46                         ((SELECT sum(value)
47                             FROM DBA_HIST_SYS_TIME_MODEL e
48                            WHERE e.SNAP_ID = &end_snap
49                              AND e.DBID = &DBID
50                              AND e.INSTANCE_NUMBER = &INST_NUM
51                              AND e.STAT_NAME = 'DB CPU') -
52                         (SELECT sum(value)
```

```
                              FROM DBA_HIST_SYS_TIME_MODEL b
                             WHERE b.SNAP_ID = &beg_snap
                               AND b.DBID = &DBID
                               AND b.INSTANCE_NUMBER = &INST_NUM
                               AND b.STAT_NAME = 'DB CPU')) / 1000000
TIME,
                         100 * ((SELECT sum(value)
                                   FROM DBA_HIST_SYS_TIME_MODEL e
                                  WHERE e.SNAP_ID = &end_snap
                                    AND e.DBID = &DBID
                                    AND e.INSTANCE_NUMBER = &INST_NUM
                                    AND e.STAT_NAME = 'DB CPU') -
                         (SELECT sum(value)
                                   FROM DBA_HIST_SYS_TIME_MODEL b
                                  WHERE b.SNAP_ID = &beg_snap
                                    AND b.DBID = &DBID
                                    AND b.INSTANCE_NUMBER = &INST_NUM
                                    AND b.STAT_NAME = 'DB CPU')) /
                         ((SELECT sum(value)
                             FROM DBA_HIST_SYS_TIME_MODEL e
                            WHERE e.SNAP_ID = &end_snap
                              AND e.DBID = &DBID
                              AND e.INSTANCE_NUMBER = &INST_NUM
                              AND e.STAT_NAME = 'DB time') -
                         (SELECT sum(value)
                             FROM DBA_HIST_SYS_TIME_MODEL b
                            WHERE b.SNAP_ID = &beg_snap
                              AND b.DBID = &DBID
                              AND b.INSTANCE_NUMBER = &INST_NUM
                              AND b.STAT_NAME = 'DB time')) PCTWTT,
                         NULL WAIT_CLASS
                    FROM DUAL
                   WHERE ((SELECT sum(value)
                             FROM DBA_HIST_SYS_TIME_MODEL e
                            WHERE e.SNAP_ID = &end_snap
                              AND e.DBID = &DBID
                              AND e.INSTANCE_NUMBER = &INST_NUM
                              AND e.STAT_NAME = 'DB CPU') -
                         (SELECT sum(value)
                             FROM DBA_HIST_SYS_TIME_MODEL b
                            WHERE b.SNAP_ID = &beg_snap
                              AND b.DBID = &DBID
```

```
                                AND b.INSTANCE_NUMBER = &INST_NUM
                                AND b.STAT_NAME = 'DB CPU')) > 0)
          ORDER BY TIME DESC, WAITS DESC)
  WHERE ROWNUM <= &TOP_N_EVENTS;
```

RAC Statistics

这一部分只在有RAC环境下才会出现，是一些全局内存中数据发送、接收方面的性能指标，还有一些全局锁的信息。除非这个数据库在运行正常是设定了一个基线作为参照，否则很难从这部分数据中直接看出性能问题。

经验

Oracle公司经验，下面GCS和GES各项指标中，凡是与时间相关的指标，只要GCS指标低于10ms，GES指标低于15ms，则一般表示节点间通讯效率正常。但是，即便时间指标正常，也不表示应用本身或应用在RAC部署中没有问题。

Global Cache Load Profile

	Per Second	Per Transaction
Global Cache blocks received:	159.23	119.37
Global Cache blocks served:	38.11	28.57
GCS/GES messages received:	516.96	387.54
GCS/GES messages sent:	654.27	490.48
DBWR Fusion writes:	0.35	0.27
Estd Interconnect traffic (KB)	1,807.48	

Global Cache Efficiency Percentages (Target local+remote 100%)

Buffer access - local cache %:	89.38
Buffer access - remote cache %:	2.26
Buffer access - disk %:	8.37

Global Cache and Enqueue Services - Workload Characteristics

Avg global enqueue get time (ms):	0.1
Avg global cache cr block receive time (ms):	1.5
Avg global cache current block receive time (ms):	0.6
Avg global cache cr block build time (ms):	0.0
Avg global cache cr block send time (ms):	0.0
Global cache log flushes for cr blocks served %:	5.3
Avg global cache cr block flush time (ms):	5.3
Avg global cache current block pin time (ms):	0.0
Avg global cache current block send time (ms):	0.0
Global cache log flushes for current blocks served %:	0.3
Avg global cache current block flush time (ms):	1.8

Global Cache and Enqueue Services - Messaging Statistics

Avg message sent queue time (ms):	0.0
Avg message sent queue time on ksxp (ms):	0.2
Avg message received queue time (ms):	0.1
Avg GCS message process time (ms):	0.0
Avg GES message process time (ms):	0.0
% of direct sent messages:	50.40
% of indirect sent messages:	48.49
% of flow controlled messages:	1.10

Global Cache Transfer Stats

		CR				Current			
Inst No	Block Class	Blocks Received	% Immed	% Busy	% Congst	Blocks Received	% Immed	% Busy	% Congst
2	data block	2,443,483	98.34	1.45	0.20	46,994,043	97.61	0.00	2.39
2	Others	4,045	99.88	0.00	0.12	6,903	99.81	0.09	0.10
2	undo header	6,791	99.09	0.82	0.09	286	98.95	0.70	0.35
2	undo block	3,942	99.72	0.20	0.08	0			

如果CR的%Busy很大，说明节点间存在大量的块争用。

Wait Events Statistics

Time Model Statistics

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	105,209.89	91.15
DB CPU	26,469.22	22.93
parse time elapsed	8,949.14	7.75
hard parse elapsed time	1,311.67	1.14
PL/SQL execution elapsed time	622.93	0.54
failed parse elapsed time	259.49	0.22
connection management call elapsed time	243.80	0.21
PL/SQL compilation elapsed time	85.76	0.07

这部分信息列出了各种操作占用的数据库时间比例。

parse time elapsed/hard parse elapsed time

通过这两个指标的对比，可以看出硬解析占整个的比例。如果很高，就说明存在大量硬解析。

% Not-Parse CPU

花费在非解析上CPU消耗占整个CPU消耗的比例。反之，则可以看出解析占用情况。如果很高，也可以反映出解析过多(进一步可以看看是否是硬解析过多)。

示例 - 计算CPU消耗

Time Model Statistics

- Total time in database user-calls (DB Time): 3254.9s
- Statistics including the word "background" measure background process time, and so do not contribute to the DB time statistic
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	3,223.44	99.03
DB CPU	1,305.89	40.12
connection management call elapsed time	27.16	0.83
parse time elapsed	10.79	0.33
PL/SQL execution elapsed time	10.14	0.31
hard parse elapsed time	1.65	0.05
hard parse (sharing criteria) elapsed time	0.96	0.03
PL/SQL compilation elapsed time	0.96	0.03
hard parse (bind mismatch) elapsed time	0.58	0.02
Java execution elapsed time	0.08	0.00
sequence load elapsed time	0.03	0.00
failed parse elapsed time	0.01	0.00
repeated bind elapsed time	0.01	0.00
DB time	3,254.90	
background elapsed time	86.54	
background cpu time	35.91	

Total DB CPU = DB CPU + background cpu time = 1305.89 + 35.91 = 1341.8 seconds

再除以总的 BUSY_TIME + IDLE_TIME

% Total CPU = 1341.8/1941.76 = 69.1%，这刚好与上面Report的值相吻合。

其实，在Load Profile部分，我们也可以看出DB对系统CPU的资源利用情况。

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	13.3	1.6	0.34	0.72
DB CPU(s):	5.3	0.7	0.14	0.29

用DB CPU per Second除以CPU Count就可以得到DB在前台所消耗的CPU%了。

这里 5.3/8 = 66.25 %

比69.1%稍小，说明DB在后台也消耗了大约3%的CPU。

Wait Class

Wait Class	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn
User I/O	10,004,348	0.00	66,093	7	86.84
Cluster	20,162,764	0.00	23,767	1	175.03
Concurrency	628,986	62.27	4,800	8	5.46
Other	1,909,838	75.38	4,350	2	16.58
System I/O	578,089	0.00	1,155	2	5.02
Application	2,901,743	0.00	682	0	25.19
Network	49,227,224	0.00	220	0	427.33

这一部分是等待的类型。可以看出那类等待占用的时间最长。

Wait Events

Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn
db file sequential read	6,944,214	0.00	48,669	7	60.28
db file scattered read	2,656,393	0.00	15,079	6	23.06
gc cr block busy	2,453,031	0.00	11,332	5	21.29
gc buffer busy	782,827	0.09	5,141	7	6.80
cursor: pin S wait on X	399,034	98.15	4,645	12	3.46
kksfbc child completion	48,486	98.81	2,438	50	0.42
db file parallel read	341,752	0.00	2,211	6	2.97

这一部分是整个实例等待事件的明细，它包含了TOP 5等待事件的信息。

%Time-outs: 超时百分比

Background Wait Events

Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn
events in waitclass Other	1,693,399	78.26	477	0	14.70
control file sequential read	187,646	0.00	471	3	1.63
log file sequential read	12,761	0.00	339	27	0.11
log file parallel write	60,215	0.00	142	2	0.52
db file parallel write	82,472	0.00	48	1	0.72
os thread startup	517	0.00	22	43	0.00

这一部分是实例后台进程的等待事件。如果我们怀疑那个后台进程(比如DBWR)无法

及时响应，可以在这里确认一下是否有后台进程等待时间过长的事件存在。

Operating System Statistics

背景知识

如果关注数据库的性能，那么当拿到一份AWR报告的时候，最想知道的第一件事情可能就是系统资源的利用情况了，而首当其冲的，就是CPU。而细分起来，CPU可能指的是：

- OS级的User%, Sys%, Idle%
- DB所占OS CPU资源的Busy%
- DB CPU又可以分为前台所消耗的CPU和后台所消耗的CPU

11g

如果数据库的版本是11g，这些信息在AWR报告中一目了然：

Host CPU (CPUs: 8 Cores: 8 Sockets: 2)

Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
2.51	10.64	75.4	2.8	0.0	21.2

Instance CPU

%Total CPU	%Busy CPU	%DB time waiting for CPU (Resource Manager)
69.1	87.7	0.0

OS级的%User为75.4，%Sys为2.8，%Idle为21.2，所以%Busy应该是78.8。
DB占了OS CPU资源的69.1，%Busy CPU则可以通过上面的数据得到:%Busy CPU = %Total CPU/(%Busy) * 100 = 69.1/78.8 * 100 = 87.69，和报告的87.7相吻合。

10g

如果是10g，则需要手工对Report里的一些数据进行计算了。Host CPU的结果来源于DBA_HIST_OSSTAT，AWR报告里已经帮忙整出了这段时间内的绝对数据(这里的时间单位是厘秒-也就是1/100秒)。

Operating System Statistics

- *TIME statistic values are diffed. All others display actual values. End Value is displayed if different
- ordered by statistic type (CPU Use, Virtual Memory, Hardware Config), Name

Statistic	Value	End Value
BUSY_TIME	152,946	
IDLE_TIME	41,230	
IOWAIT_TIME	16	
NICE_TIME	0	
SYS_TIME	5,462	
USER_TIME	146,355	
LOAD	3	11
PHYSICAL_MEMORY_BYTES	33,753,260,032	
NUM_CPUS	8	
NUM_CPU_CORES	8	
NUM_CPU_SOCKETS	2	
GLOBAL_RECEIVE_SIZE_MAX	4,194,304	
GLOBAL_SEND_SIZE_MAX	2,097,152	
TCP_RECEIVE_SIZE_DEFAULT	87,380	
TCP_RECEIVE_SIZE_MAX	4,194,304	
TCP_RECEIVE_SIZE_MIN	4,096	
TCP_SEND_SIZE_DEFAULT	16,384	
TCP_SEND_SIZE_MAX	4,194,304	
TCP_SEND_SIZE_MIN	4,096	

%User =USER_TIME/(BUSY_TIME+IDLE_TIME)*100 = 146355/(152946+41230)*100 = 75.37

%Sys =SYS_TIME/(BUSY_TIME+IDLE_TIME)*100*

*%Idle =IDLE_TIME/(BUSY_TIME+IDLE_TIME)*100

ELAPSED_TIME

这里已经隐含着这个AWR报告所捕捉的两个snapshot之间的时间长短了。有下面的公式。正确的理解这个公式可以对系统CPU资源的使用及其度量的方式有更深一步的理解。

BUSY_TIME + IDLE_TIME = ELAPSED_TIME * CPU_COUNT

推算出：ELAPSED_TIME = (152946+41230)/8/100 = 242.72 seconds

SQL Statistics

SQL ordered by Elapsed Time

Elapsed Time (s)	CPU Time (s)	Executions	Elap per Exec (s)	% Total DB Time	SQL Id	SQL Module	SQL Text
9,570	4,559	5,597,518	0.00	8.29	6aupq5kk4czbj		select SERIAL_ID, PRD_ID, PRD_...
4,642	1,232	1,984,557	0.00	4.02	4dubw0d8qj9q7		select SEQ_ID, SERIAL_ID, CREA...
3,536	1,041	1,419,241	0.00	3.06	au684ytxv0spg	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_...
3,224	2,868	69,116	0.05	2.79	dbxbabg2gusva	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_...
2,339	328	0		2.03	7b10snzfvt0pn	SQL*Plus	insert into t_tmp_prd_cls111 s...
1,574	269	1	1573.81	1.36	bvzq78mwd1r91	SQL*Plus	insert into t_tmp_prd_cls111 s...

这一部分是按照SQL执行时间从长到短的排序。

Elapsed Time(S)

SQL语句执行用总时长，此排序就是按照这个字段进行的。注意该时间不是单个SQL跑的时间，而是监控范围内SQL执行次数的总和时间。单位时间为秒。Elapsed Time = CPU Time + Wait Time

CPU Time(s)

为SQL语句执行时CPU占用时间总时长，此时间会小于等于Elapsed Time时间。单位时间为秒。

Executions

SQL语句在监控范围内的执行次数总计。如果Executions=0，则说明语句没有正常完成，被中间停止，需要关注。

Elap per Exec(s)

执行一次SQL的平均时间。单位时间为秒。

% Total DB Time

为SQL的Elapsed Time时间占数据库总时间的百分比。

SQL ID

SQL语句的ID编号，点击之后就能导航到下边的SQL详细列表中，点击IE的返回可以回到当前SQL ID的地方。

SQL Module

显示该SQL是用什么方式连接到数据库执行的，如果是用SQL*Plus或者PL/SQL链接上来的那基本上都是有人在调试程序。一般用前台应用链接过来执行的sql该位置为空。

SQL Text

简单的SQL提示，详细的需要点击SQL ID。
分析说明
如果看到SQL语句执行时间很长，而CPU时间很少，则说明SQL在I/O操作时(包括逻辑I/O和物理I/O)消耗较多。可以结合前面I/O方面的报告以及相关等待事件，进一步分析是否是I/O存在问题。当然SQL的等待时间主要发生在I/O操作方面，不能说明系统就存在I/O瓶颈，只能说SQL有大量的I/O操作。
如果SQL语句执行次数很多，需要关注一些对应表的记录变化。如果变化不大，需要从前面考虑是否大多数操作都进行了Rollback，导致大量的无用功。

SQL ordered by CPU Time

CPU Time (s)	Elapsed Time (s)	Executions	CPU per Exec (s)	% Total DB Time	SQL Id	SQL Module	SQL Text
4,559	9,570	5,597,518	0.00	8.29	6augq5kk4czbj		select SERIAL_ID, PRD_ID, PRD_...
2,868	3,224	69,116	0.04	2.79	ddxbabq2gusva	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_...
1,232	4,642	1,984,557	0.00	4.02	4dubw0d8qj9q7		select SEQ_ID, SERIAL_ID, CREA...
1,041	3,536	1,419,241	0.00	3.06	au68dytxv0spg	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_...
667	1,332	2	333.33	1.15	6m6wrfzr8m4qv	perl@dw132 (TNS V1-V3)	SELECT SERIAL_ID, PRD_ID, SERI...
568	1,029	1	568.31	0.89	d4a337bvgftw3	perl@dw132 (TNS V1-V3)	SELECT SERIAL_ID, PRD_ID, SERI...

记录了执行占CPU时间总和时间最长的TOP SQL(请注意是监控范围内该SQL的执行占CPU时间总和，而不是单次SQL执行时间)。这部分是SQL消耗的CPU时间从高到底的排序。

CPU Time (s)

SQL消耗的CPU时间。

Elapsed Time (s)

SQL执行时间。

Executions

SQL执行次数。

CPU per Exec (s)

每次执行消耗CPU时间。

% Total DB Time

SQL执行时间占总共DB time的百分比。

SQL ordered by Gets

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
198,769,883	69,116	2,875.89	32.64	2868.23	3224.31	ddxbabg2gusva	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) ,to_...
33,883,758	5,597,518	6.05	5.56	4559.25	9570.19	6aupq5kk4czbj		select SERIAL_ID, PRD_ID, PRD_...
18,517,867	69,116	267.92	3.04	317.54	798.61	8tnswu7uj9v89	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) ,to_...
17,794,379	5,916,634	3.01	2.92	136.53	139.17	5yqpdz9jdcj22	push@linux (TNS V1-V3)	select PRD_SUM into :b1:b2 fr...
16,553,931	12,283	1,347.71	2.72	252.62	1207.18	1q5294cnvv7s3	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) ,to_...
15,703,227	22,914	685.31	2.58	296.63	1172.56	79raj8xn1vbu0	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) ,to_...

这部分列出SQL获取的内存数据块的数量，按照由大到小的顺序排序。buffer get其实就是逻辑读或一致性读。在sql 10046里面，也叫query read。表示一个语句在执行期间的逻辑IO，单位是块。在报告中，该数值是一个累计值。Buffer Get=执行次数 * 每次的buffer get。记录了执行占总buffer gets(逻辑IO)的TOP SQL(请注意是监控范围内该SQL的执行占Gets总和，而不是单次SQL执行所占的Gets)。

Buffer Gets

SQL执行获得的内存数据块数量。

Executions

SQL执行次数。

Gets per Exec

每次执行获得的内存数据块数量。

%Total

占总数的百分比。

CPU Time (s)

消耗的CPU时间。

Elapsed Time (s)

SQL执行时间。

筛选SQL的标准

因为statspack/awr列出的是总体的top buffer，它们关心的是总体的性能指标，而不是把重心放在只执行一次的语句上。为了防止过大，采用了以下原则。如果有sql没有使用绑定变量，执行非常差但是由于没有绑定，因此系统人为是不同的sql。有可能不会被列入到这个列表中。

大于阀值buffer_gets_th的数值，这是sql执行缓冲区获取的数量(默认10000)。

小于define top_n_sql=65的数值。

SQL ordered by Reads

Physical Reads	Executions	Reads per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
4,265,218	0		8.37	328.13	2339.28	7b10snzfvt0pn	SQL*Plus	insert into t_tmp_prd_cls111 s...
3,805,656	1	3,805,656.00	7.47	268.64	1573.81	bvzcq78mvd1r91	SQL*Plus	insert into t_tmp_prd_cls111 s...
3,741,580	0		7.34	189.38	1559.04	2qu30krm4z079	SQL Developer	create table oddnow_tmp_prd_cl...
3,001,784	0		5.89	215.95	1345.37	dj8byt9nhu5aa	SQL*Plus	insert into t_tmp_prd_cls111 s...

这部分列出了SQL执行物理读的信息，按照从高到低的顺序排序。记录了执行占总磁盘物理读(物理IO)的TOP SQL(请注意是监控范围内该SQL的执行占磁盘物理读总和，而不是单次SQL执行所占的磁盘物理读)。

Physical Reads

SQL物理读的次数。

Executions

SQL执行次数。

Reads per Exec

SQL每次执行产生的物理读。

%Total

占整个物理读的百分比。

CPU Time (s)

SQL执行消耗的CPU时间。

Elapsed Time (s)

SQL的执行时间。

SQL ordered by Executions

Executions	Rows Processed	Rows per Exec	CPU per Exec (s)	Elap per Exec (s)	SQL Id	SQL Module	SQL Text
5,916,634	43,862	0.01	0.00	0.00	5yqpdz9jdcj22	push@linux (TNS V1-V3)	select PRD_SUM into :b1:b2 fr...
5,597,518	5,595,135	1.00	0.00	0.00	6aupq5kk4czbj		select SERIAL_ID, PRD_ID, PRD_...
4,315,347	30,609	0.01	0.00	0.00	c3dydzh2rqggc		select PRD_SUM from prd_user P...
1,984,557	646,166	0.33	0.00	0.00	4dubw0d8qj9q7		select SEQ_ID, SERIAL_ID, CREA...

这部分列出了SQL执行次数的信息，按照从大到小的顺序排列。如果是OLTP系统的话，这部分比较有用。因此SQL执行频率非常大，SQL的执行次数会对性能有比较大的影响。OLAP系统因为SQL重复执行的频率很低，因此意义不大。

Executions

SQL的执行次数。

Rows Processed

SQL处理的记录数。

Rows per Exec

SQL每次执行处理的记录数。

CPU per Exec (s)

每次执行消耗的CPU时间。

Elap per Exec (s)

每次执行的时长。

SQL ordered by Parse Calls

Parse Calls	Executions	% Total Parses	SQL Id	SQL Module	SQL Text
5,597,470	5,597,518	35.59	6aupq5kk4czbj		select SERIAL_ID, PRD_ID, PRD_...
4,315,197	4,315,347	27.44	c3dydzh2rqggc		select PRD_SUM from prd_user.P...
1,984,558	1,984,557	12.62	4dubw0d8qj9q7		select SEQ_ID, SERIAL_ID, CREA...
246,157	246,162	1.57	5n3pvxzjdwsa6		select PRD_SUM from prd_user.P...
207,199	207,210	1.32	0h6b2sajwb74n		select privilege#, level from ...
88,693	88,693	0.56	0pm26413q1tb2	bidorder_test@linux (TNS V1-V3)	UPDATE PRD_BID_RECORD SET SERI...

这部分列出了SQL按分析次(软解析)数的信息，按照从高到底的顺序排列。这部分对OLTP系统比较重要，这里列出的总分析次数并没有区分是硬分析还是软分析。但是即使是软分析，次数多了，也是需要关注的。这样会消耗很多内存资源，引起latch的等待，降低系统的性能。软分析过多需要检查应用上是否有频繁的游标打开、关闭操作。

Parse Calls

SQL分析的次数。

Executions

SQL执行的次数。

% Total Parses

占整个分析次数的百分比。

SQL ordered by Sharable Memory

记录了SQL占用library cache的大小的TOP SQL。

Sharable Mem (b)

占用library cache的大小。单位是byte。

SQL ordered by Version Count

Version Count	Executions	SQL Id	SQL Module	SQL Text
29	109	5jxwda6s8nvc7		select tablespace_id, rfno, ...

这部分列出了SQL多版本的信息。记录了SQL的打开子游标的TOP SQL。一个SQL产生多版本的原因有很多，可以查询视图v$sql_sahred_cursor视图了解具体原因。对于OLTP系统，这部分值得关注，了解SQL被重用的情况。

Version Count

SQL的版本数。

Executions

SQL的执行次数。

SQL ordered by Cluster Wait Time

Cluster Wait Time (s)	CWT % of Elapsd Time	Elapsed Time (s)	CPU Time (s)	Executions	SQL Id	SQL Module	SQL Text
6,810.07	71.16	9,570.19	4,559.25	5,597,518	6aupq5kk4czbj		select SERIAL_ID, PRD_ID, PRD
4,358.46	93.90	4,641.55	1,232.47	1,984,557	4dubw9d8gj9q7		select SEQ_ID, SERIAL_ID, CRE
3,389.51	95.85	3,536.28	1,041.01	1,419,241	au684ytxv9spg	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_
1,088.28	72.95	1,491.71	468.16	36,233	7kj4ajb341pqz	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_
919.92	78.45	1,172.56	296.63	22,914	79raj8xn1vhu9	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_
644.50	53.39	1,207.18	252.62	12,283	1q5294cnvv7s3	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_
615.80	77.11	798.61	317.54	69,116	8tnswu7uj9v89	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_
296.81	9.21	3,224.31	2,868.23	69,116	dbxbabq2susva	push@linux (TNS V1-V3)	select to_char(T.SEQ_ID) , to_
277.08	63.45	436.70	301.97	677,190	g8tta211my8sy	push@linux (TNS V1-V3)	select to_char(P.SERIAL_ID) ,

记录了集群的等待时间的TOP SQL。这部分只在RAC环境中存在，列出了实例之间共享内存数据时发生的等待。在RAC环境下，几个实例之间需要有一种锁的机制来保证数据块版本的一致性，这就出现了一类新的等待事件，发生在RAC实例之间的数据访问等待。对于RAC结构，还是采用业务分隔方式较好。这样某个业务固定使用某个实例，它访问的内存块就会固定地存在某个实例的内存中，这样降低了实例之间的GC等待事件。此外，如果RAC结构采用负载均衡模式，这样每个实例都会被各种应用的会话连接，大量的数据块需要在各个实例的内存中被拷贝和锁定，会加剧GC等待事件。

Cluster Wait Time (s)

集群等待时长。

CWT % of Elapsd Time

集群操作等待时长占总时长的百分比。

Elapsed Time(s)

SQL执行总时长。

Complete List of SQL Text

SQL Id	SQL Text
02ytyvk93hb45	insert into t_tmp_prd_cls111 select * from product_ora partition (PART_PRODUCT_200804) where substr(to_char(cls_id), 1, 3)='111'
0cx6xjyx8cywa	select count(1) from PRD_USER.PRODUCT_ORA where CREATE_TIME >= 1214755200 and CREATE_TIME < 1214841600
0h6b2sajwb74n	select privilege#, level from sysauth$ connect by grantee#=prior privilege# and privilege#>0 start with grantee#=:1 and privilege#>0
0k8522nmdzg4k	select privilege# from sysauth$ where (grantee#=:1 or grantee#=1) and privilege#>0
0pm26413qttb2	UPDATE PRD_BID_RECORD SET SERIAL_ID=:a, PRD_ID=:b, BUYER_ONLYID=:c, BUYER_LOGINID=:d, BID_PRD_NUM=:e, REAL_VALUE=:f, REAL_PRD_NUM=:g, PRD_LEADER BID_ORDER_FLAG=:i, CTOC_ORDER_ID=:j, ORDER_CREATE_TIME=unix_to_oracle(:k), UPDATE_TIME=:l WHERE BID_ORDER_ID=:m
12yj7g0qfcppj	select P.SERIAL_ID from PRODUCT_ORA P where (((P.USR_ONLYID=:b1 and P.SERIAL_FLAG=1) and P.PRD_FLAG=5) and P.SAL_END_TIME>=:b2)
1ang6ud51th44	select PRD_SUM into :b1,:b2 from PRD_PROP P where P.PRD_ID=:b3

这部分是上面各部分涉及的SQL的完整文本。

Instance Activity Statistics

Instance Activity Stats

Statistic	Total	per Second	per Trans
CPU used by this session	1,813,626	21.00	15.74
CPU used when call started	1,809,868	20.96	15.71
CR blocks created	26,470	0.31	0.23
Cached Commit SCN referenced	17,648	0.20	0.15
Commit SCN cached	16	0.00	0.00
DB time	37,783,580	437.52	327.99
DBWR checkpoint buffers written	994,131	11.51	8.63
DBWR checkpoints	144	0.00	0.00
DBWR fusion writes	30,588	0.35	0.27
DBWR object drop buffers written	0	0.00	0.00
DBWR parallel query checkpoint buffers written	0	0.00	0.00
DBWR tablespace checkpoint buffers written	16	0.00	0.00
DBWR thread checkpoint buffers written	6,697	0.08	0.06
DBWR transaction table writes	2,389	0.03	0.02
DBWR undo block writes	98,009	1.13	0.85
DFO trees parallelized	146	0.00	0.00
PX local messages recv'd	12	0.00	0.00
PX local messages sent	12	0.00	0.00

这部分是实例的信息统计，项目非常多。对于RAC架构的数据库，需要分析每个实例的AWR报告，才能对整体性能做出客观的评价。

CPU used by this session

这个指标用来上面在当前的性能采集区间里面，Oracle消耗的CPU单位。一个CPU单位是1/100秒。从这个指标可以看出CPU的负载情况。

案例 - 分析系统CPU繁忙程度

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
db file sequential read	6,944,214	48,669	7	42.2	User I/O
CPU time		26,469		22.9	

在TOP5等待事件里，找到"CPU time"，可以看到系统消耗CPU的时间为26469秒。

Statistic	Total	per Second	per Trans
CPU used by this session	1,813,626	21.00	15.74

在实例统计部分，可以看到整个过程消耗了1813626个CPU单位。每秒钟消耗21个CPU单位，对应实际的时间就是0.21秒。也就是每秒钟CPU的处理时间为0.21秒。

NUM_CPUS	8

系统内CPU个数为8。每秒钟每个CPU消耗是21/8=2.6(个CPU单位)。在一秒钟内，每个CPU处理时间就是2.6/100=0.026秒。

- 总体来看，当前数据库每秒钟每个CPU处理时间才0.026秒，远远算不上高负荷。数据库CPU资源很丰富，远没有出现瓶颈。

IO Stats

Tablespace IO Stats

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
TBS_PRD_200806	9,005,318	104	4.86	2.38	26,587	0	146,076	3.80
TBS_PRD_200805	972,968	11	2.02	6.07	8,974	0	727	5.97
TBS_PRD_200807	862,571	10	6.29	1.03	846	0	132,087	2.55
TBS_PRD_200804	768,923	9	2.34	6.85	1,977	0	86	3.02
TBS_PRD_200803	615,823	7	2.78	8.25	873	0	9	0.00
TBS_PRD_IDX	582,720	7	8.39	1.00	29,909	0	371,948	8.44
TBS_PRD_200801	460,402	5	3.13	9.57	412	0	7	0.00
TBS_PRD_200712	433,162	5	3.25	10.35	541	0	3	0.00
TBS_PRD_200802	326,105	4	2.89	8.68	941	0	6	0.00

表空间的I/O性能统计。

Reads

发生了多少次物理读。

Av Reads/s

每秒钟物理读的次数。

Av Rd(ms)

平均一次物理读的时间(毫秒)。一个高相应的磁盘的响应时间应当在10ms以内，最好不要超过20ms；如果达到了100ms，应用基本就开始出现严重问题甚至不能正常运行。

Av Blks/Rd

每次读多少个数据块。

Writes

发生了多少次写。

Av Writes/s

每秒钟写的次数。

Buffer Waits

获取内存数据块等待的次数。

Av Buf Wt(ms)

获取内存数据块平均等待时间。

File IO Stats

Tablespace	Filename	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf W
EXAMPLE	+ASM_DISK2/prddb/datafile/example.276.632944223	28	0	0.36	1.00	27	0	0	
SYSAUX	+ASM_DISK2/prddb/datafile/sysaux.269.632944153	16,961	0	2.93	1.01	9,338	0	0	
SYSTEM	+ASM_DISK2/prddb/datafile/system.268.632944153	4,169	0	7.36	1.09	1,001	0	3	
TBS_DATA	+ASM_DISK2/prddb/datafile/tbs_data.304.656646689	28	0	0.00	1.00	27	0	0	
TBS_ENDITEM_IDX	+ASM_DISK2/prddb/datafile/tbs_enditem_idx.321.633004	64	0	0.16	1.00	25	0	0	
TBS_ENDITEM_MAX	+ASM_DISK2/prddb/datafile/tbs_enditem_max.379.633005	64	0	0.16	1.00	25	0	0	
TBS_GCARD_DATA	+ASM_DISK2/prddb/datafile/tbs_gcard_data.377.6443462	64	0	0.00	1.00	25	0	0	
TBS_GCARD_IDX	+ASM_DISK2/prddb/datafile/tbs_gcard_idx.376.64434627	64	0	0.16	1.00	25	0	0	
TBS_PRD_200712	+ASM_DISK2/prddb/datafile/tbs_prd_200712.305.6330045	341,244	4	3.26	10.32	165	0	2	
TBS_PRD_200712	+ASM_DISK2/prddb/datafile/tbs_prd_200712.378.6422481	91,918	1	3.22	10.44	376	0	1	
TBS_PRD_200801	+ASM_DISK2/prddb/datafile/tbs_prd_200801.306.6330045	339,241	4	3.22	9.87	132	0	7	
TBS_PRD_200801	+ASM_DISK2/prddb/datafile/tbs_prd_200801.375.6447166	121,161	1	2.86	8.74	280	0	0	
TBS_PRD_200802	+ASM_DISK2/prddb/datafile/tbs_prd_200802.307.6330045	161,042	2	2.86	8.62	636	0	4	

文件级别的I/O统计。

Advisory Statistics

顾问信息。这块提供了多种顾问程序，提出在不同情况下的模拟情况。包括databuffer、pga、shared pool、sga、stream pool、java pool等的情况。

Buffer Pool Advisory

P	Size for Est (M)	Size Factor	Buffers for Estimate	Est Phys Read Factor	Estimated Physical Reads
D	752	0.10	90,193	1.10	514,586,367
D	1,504	0.20	180,386	1.07	500,823,725
D	2,256	0.30	270,579	1.06	492,617,472
D	3,008	0.40	360,772	1.04	485,481,748
D	3,760	0.50	450,965	1.03	479,829,339
D	4,512	0.59	541,158	1.02	475,680,569
D	5,264	0.69	631,351	1.01	472,281,144
D	6,016	0.79	721,544	1.01	469,893,703
D	6,768	0.89	811,737	1.00	467,877,572
D	7,520	0.99	901,930	1.00	466,141,913
D	7,584	1.00	909,606	1.00	465,999,917
D	8,272	1.09	992,123	1.00	464,585,308
D	9,024	1.19	1,082,316	0.99	463,341,070
D	9,776	1.29	1,172,509	0.99	462,079,847

Buffer pool的大小建议。

Size for Est (M)

Oracle估算Buffer pool的大小。

Size Factor

估算值与实际值的比例。如果0.9就表示估算值是实际值的0.9倍。1.0表示buffer pool的实际大小。

Buffers for Estimate

估算的Buffer的大小(数量)。

Est Phys Read Factor

估算的物理读的影响因子，是估算物理读和实际物理读的一个比例。1.0表示实际的物理读。

Estimated Physical Reads

估计的物理读次数。

PGA Memory Advisory

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count
461	0.13	95,850.74	36,126.25	73.00	0
922	0.25	95,850.74	32,166.38	75.00	0
1,843	0.50	95,850.74	26,983.44	78.00	0
2,765	0.75	95,850.74	26,983.44	78.00	0
3,686	1.00	95,850.74	26,983.44	78.00	0
4,423	1.20	95,850.74	9,892.64	91.00	0
5,160	1.40	95,850.74	9,892.64	91.00	0
5,898	1.60	95,850.74	9,892.64	91.00	0
6,635	1.80	95,850.74	9,892.64	91.00	0
7,372	2.00	95,850.74	9,892.64	91.00	0

PGA的大小建议。

PGA Target Est (MB)

PGA的估算大小。

Size Factr

影响因子，作用和buffer pool advisory中相同。

W/A MB Processed

Oracle为了产生影响估算处理的数据量。

Estd Extra W/A MB Read/ Written to Disk

处理数据中需要物理读写的数据量。

Estd PGA Cache Hit %

估算的PGA命中率。

Estd PGA Overalloc Count

需要在估算的PGA大小下额外分配内存的个数。

Shared Pool Advisory

Shared Pool Size (M)	SP Size Factr	Est LC Size (M)	Est LC Mem Obj	Est LC Time Saved (s)	Est LC Time Saved Factr	Est LC Load Time (s)	Est LC Load Time Factr	Est LC Mer Hits
1,136	0.49	264	18,267	18,860,797	0.98	335,522	13.42	465,
1,376	0.59	503	19,584	18,961,376	0.99	234,943	9.40	465,
1,616	0.69	742	20,928	19,047,526	0.99	148,793	5.95	465,
1,856	0.79	981	22,292	19,104,468	1.00	91,851	3.67	465,
2,096	0.90	1,220	23,681	19,147,262	1.00	49,057	1.96	465,
2,336	1.00	1,459	25,101	19,171,318	1.00	25,001	1.00	465,
2,576	1.10	1,698	26,479	19,186,648	1.00	9,671	0.39	465,

建议器通过设置不同的共享池大小，来获取相应的性能指标值。

Shared Pool Size(M)

估算的共享池大小。

SP Size Factr

共享池大小的影响因子。

Est LC Size (M)

估算的库高速缓存占用的大小。

Est LC Mem Obj

高速缓存中的对象数。

Est LC Time Saved (s)

需要额外将对象读入共享池的时间。

Est LC Time Saved Factr

对象读入共享池时间的影响因子。
表示每一个模拟的shared pool大小对重新将对象读入共享池的影响情况。当这个值的变化很小或不变的时候，增加shared pool的大小就意义不大。

Est LC Load Time (s)

分析所花费的时间。

Est LC Load Time Factr

分析花费时间的影响因子。

Est LC Mem Obj Hits

内存中对象被发现的次数。

SGA Target Advisory

SGA Target Size (M)	SGA Size Factor	Est DB Time (s)	Est Physical Reads
2,500	0.25	1,839,219	512,506,670
5,000	0.50	1,622,957	490,651,276
7,500	0.75	1,555,098	473,782,080
10,000	1.00	1,528,352	465,999,882
12,500	1.25	1,508,483	460,221,483

建议器对SGA整体的性能的一个建议。

SGA Target Size (M)

估算的SGA大小。

SGA Size Factor

SGA大小的影响因子。

Est DB Time (s)

估算的SGA大小计算出的DB Time。

Est Physical Reads

物理读的次数。

Latch Statistics

Latch Activity

Latch Name	Get Requests	Pct Get Miss	Avg Slps /Miss	Wait Time (s)	NoWait Requests	Pct NoWait Miss
ASM allocation	11,470	0.00		0	0	
ASM db client latch	131,281	0.00		0	0	
ASM map headers	14,914	0.00		0	0	
ASM map load waiting list	4,464	0.00		0	0	
ASM map operation freelist	14,396	0.05	0.00	0	0	

Get Requests/Pct Get Miss/Avg Slps /Miss

表示愿意等待类型的latch的统计信息。

NoWait Requests/Pct NoWait Miss

表示不愿意等待类型的latch的统计信息。

Pct Misses

比例最好接近0。

Segment Statistics

Segments by Logical Reads

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Logical Reads	%Total
PRD_USER	TBS_TASK_DATA	PRDBATCH_TASK		TABLE	225,625,776	37.05
PRD_USER	TBS_PRD_IDX	PK_PRODUCT_ORA_SERIALID		INDEX	176,723,328	29.02
PRD_USER	TBS_PRD_200806	PRODUCT_ORA	UCT_200806	TABLE PARTITION	51,998,256	8.54
PRD_USER	TBS_PRD_IDX	PK_PRD_SUM		INDEX	32,857,744	5.40
PRD_USER	TBS_PRD_200807	PRODUCT_ORA	UCT_200807	TABLE PARTITION	23,857,232	3.92

段的逻辑读情况。

Segments by Physical Reads

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Physical Reads	%Total
PRD_USER	TBS_PRD_200806	PRODUCT_ORA	UCT_200806	TABLE PARTITION	20,206,460	39.64
PRD_USER	TBS_PRD_200805	PRODUCT_ORA	UCT_200805	TABLE PARTITION	5,021,647	9.85
PRD_USER	TBS_PRD_200804	PRODUCT_ORA	UCT_200804	TABLE PARTITION	4,530,107	8.89
PRD_USER	TBS_PRD_200803	PRODUCT_ORA	UCT_200803	TABLE PARTITION	4,373,575	8.58
PRD_USER	TBS_PRD_200712	PRODUCT_ORA	UCT_200712	TABLE PARTITION	3,969,633	7.79

段的物理读情况。

Segments by Buffer Busy Waits

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Buffer Busy Waits	% of Capture
PRD_USER	TBS_PRD_IDX	PK_PRODUCT_ORA_SERIALID		INDEX	162	98.18
PRD_USER	TBS_PRD_IDX	PK_PRD_SUM		INDEX	2	1.21
PRD_USER	TBS_PRD_DATA	PRD_SUM		TABLE	1	0.61

从这部分可以发现那些对象访问频繁。Buffer Busy Waits事件通常由于某些数据块太过频繁的访问，导致热点块的产生。

Segments by Row Lock Waits

AWR报告Segment Statistics部分的Segments by Row Lock Waits，非常容易引起误解，它包含的不仅仅是事务的行级锁等待，还包括了索引分裂的等待。之前我一直抱怨为什么v$segment_statistics中没有统计段级别的索引分裂计数，原来ORACLE已经实现了。但是统计进这个指标中，你觉得合适吗?

其他问题

SQL运行周期对报告的影响

对SQL语句来讲，只有当它执行完毕之后，它的相关信息才会被Oracle所记录(比如：CPU时间、SQL执行时长等)。当时当某个SQL终止于做AWR报告选取的2个快照间隔时间之后，那么它的信息就不能被这个AWR报告反映出来。尽管它在采样周期里面的运行，也消耗了很多资源。
也就是说某个区间的性能报告并不能精确地反映出在这个采样周期中资源的消耗情

况。因为有些在这个区间运行的SQL可能结束于这个时间周期之后，也可能有一些SQL在这个周期开始之前就已经运行了很久，恰好结束于这个采样周期。这些因素都导致了采样周期里面的数据并不绝对是这个时间段发生的所有数据库操作的资源使用的数据。