如何分析 AWR 报告

中国 Oracle 用户组

作者：黄凯耀(Kaya)

http://www.acoug.org

版本	发布时间
1.0	2011/3/14

目录

概述：本篇文章重点对 AWR 报告中的 DB Time、DB CPU、IO 等数据进行了说明，可帮助读者更加清楚的理解这些数据代表的含义，与数据库的性能表现有何关系。同时通过两个简短的例子，实践如何分析 AWR 报告。

1 AWR 概述

Automatic Workload Repository(AWR)是 10g 引入的一个重要组件。在里面存贮着近期一段时间内（默认是 7 天）数据库活动状态的详细信息。

AWR 报告是对 AWR 视图进行查询而得到的一份自动生成的报告。可以通过下面的脚本手工得到一份 AWR 报告。

```
exec dbms_workload_repository.create_snapshot;

... running the specified workload

exec dbms_workload_repository.create_snapshot;

@?/rdbms/admin/awrrpt
```

通过 AWR 和 AWR 报告，DBA 可以容易地获知最近数据库的活动状态，数据库的各种性能指标的变化趋势曲线，最近数据库可能存在的异常，分析数据库可能存在的性能瓶颈从而对数据库进行优化。

AWR 报告所有的数据来源于 AWR 视图，即以 DBA_HIST_开头的所有系统表，Database Reference 有对所有这些系统表的描述，这应该是 Oracle 官方对 AWR 报告的官方注释了。而对于如何有效地去分析 AWR 报告，这可能更需要 DBA 经验的日积月累。

AWR 的前身是 Statspack，Statspack 在 10g 和 11g 中也有提供，同时和 AWR 一起做了同步更新，而且 Statspack 是公开源代码的，因此，关于 Statspack 的资料，还有 Statspack 的源代码，都是理解 AWR 的一个有用的辅助。

本文着重对 AWR 中的一些要点进行剖析，欢迎一起讨论 AWR 相关的问题。

2 DB CPU - CPU 负载分析

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：

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

如果数据库的版本是 11g，那么很幸运的，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

分析上面的图片，我们可以得出下面的结论：

- OS 级的 User%，Sys%，Idle%：

OS 级的%User 为 75.4，%Sys 为 2.8，%Idle 为 21.2，所以%Busy 应该是 100-21.1=78.8。

- DB 所占 OS CPU 资源的 Busy%

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。

如果是 10g 呢，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 centi second,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	

根据上面的数据，稍加计算分析便可得出下面的结果。

- OS 级的 User%，Sys%， Idle%：

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

%Sys = SYS_TIME/ (BUSY_TIME+IDLE_TIME)*100=5462/ (152946+41230)*100=2.81

%Idle = IDLE_TIME/ (BUSY_TIME+IDLE_TIME)*100=21230/ (152946+41230)*100=10.93

值得注意的，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：

BUSY_TIME + IDLE_TIME = ELAPSED_TIME * CPU_COUNT

注意：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。

因此 ELAPSED_TIME = (152946+41230)/8/100 = 242.72 seconds

至于 DB 对 CPU 的利用情况，这就涉及到 10g 新引入的一个关于时间统计的视图 V$SYS_TIME_MODEL，简单而言，Oracle 采用了一个统一的时间模型对一些重要的时间指标进行了记录，具体而言，这些指标包括：

1) Background elapsed time

2) **Background CPU time**

3) RMAN CPU time (backup/restore)

1) DB time

2) **DB CPU**

2) Connection management call elapsed time

2) Sequence load elapsed time

2) SQL execute elapsed time

2) Parse time elapsed

3) Hard parse elapsed time

4) Hard parse (sharing criteria) elapsed time

5) Hard parse (bind mismatch) elapsed time

3) Failed parse elapsed time

4) Failed parse (out of shared memory) elapsed time

2) PL/SQL execution elapsed time

2) Inbound PL/SQL RPC elapsed time

2) PL/SQL compilation elapsed time

2) Java execution elapsed time

2) Repeated bind elapsed time

这里我们关注的只有和 CPU 相关的两个： Background CPU time 和 DB CPU。

这两个值在 AWR 里面也有记录：

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
Total DB CPU 除以总的 BUSY_TIME + IDLE_TIME 可得出% Total CPU。
% 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，这是不是一个最简单的方法了呢？

3 DB Time – 进程消耗时间分析

DB CPU 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 全部处于繁忙状态的话，一秒钟内的 DB CPU 就是 N 秒。

如何去表征一个系统的繁忙程度呢？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 一节就有了对 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

这个系统的 CPU 个数是 8，因此我们可以知道前台进程用了系统 CPU 的 7.1/8=88.75%。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 CPU 占 DB Time 的比重呢？ 7.1/11.7= 60.68%

Top 5 Timed Events，或许很多人都对它有所耳闻，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

注意到，我们刚刚已经算出了 DB CPU 的%DB time 为 60%左右。

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。

回过头再研究下这个 Top 5 Timed Foreground Events，如果先不看 Load Profile，你能说出这个一个 CPU-Bound 的工作负载吗？

答案是否定的，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。否则，系统可以是一个很 IDLE 的系统呢。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。

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：

Foreground Wait Class

- s - second, ms - millisecond - 1000th of a second
- ordered by wait time desc, waits desc
- %Timeouts: value of 0 indicates value was < .5%. Value of null is truly 0
- Captured Time accounts for 71.5% of Total DB time 10,711.20 (s)
- Total FG Wait Time: 1,182.63 (s) DB CPU time: 6,474.65 (s)
  - 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？这里其实又隐含着一个 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了。

不要忘了在这个例子中，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-Bound 的系统。

4 IO 数据分析

除了 DB CPU、DB Time，或许另一个比较常用的指标应该是 IO 的利用情况。关于 IO 的指标就比较多了，单单在 Load Profile 里面就有 5 个，在 DB Time 和 DB CPU 的下面：

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	13.9	61.3	1.14	0.44
DB CPU(s):	0.9	4.0	0.07	0.03
Redo size:	5,597.7	24,762.8		
Logical reads:	197,031.8	871,617.5		
Block changes:	13.4	59.4		
Physical reads:	196,271.4	868,253.7		
Physical writes:	2.0	8.7		
User calls:	31.2	137.9		
Parses:	10.8	47.5		
Hard parses:	0.2	0.9		
W/A MB processed:	186,700.1	825,912.9		
Logons:	5.5	24.4		
Executes:	12.1	53.7		
Rollbacks:	0.1	0.5		
Transactions:	0.2			

这 5 个指标的值都来自 V$SYSTAT 视图，分别是：

- Redo Size: ‘redo size’

- Logical reads = 'session logical reads' or ('db block gets' + 'consistent gets')
- Blocks Changes = 'db block changes'
- Physical reads = 'physical reads'
- Physical writes = 'physical writes'

具体指标的解释可以参考 Database Reference

如何得到系统大致的 MBPS(Megabits Per Second)呢？

MBPS= (Physical reads + Physical writes) * Block_Size = (196,271.4+2.0)*8*1024/1024/1024 = 1533 MB/s

更准确的 MBPS 可以从 Instance Activity Stats 部分获得。

Statistic	Total	per Second	per Trans
physical IO disk bytes	576,202,341,376	1,608,052,905.83	7,113,609,152.79
physical read IO requests	577,782	1,612.46	7,133.11
physical read bytes	576,131,481,600	1,607,855,151.92	7,112,734,340.74
physical read total IO requests	580,801	1,620.89	7,170.38
physical read total bytes	576,180,494,336	1,607,991,935.59	7,113,339,436.25
physical read total multi block requests	565,191	1,577.32	6,977.67
physical reads	70,328,550	196,271.38	868,253.70
physical reads cache	110	0.31	1.36
physical reads cache prefetch	42	0.12	0.52
physical reads direct	70,328,440	196,271.07	868,252.35
physical reads direct (lob)	0	0.00	0.00
physical reads direct temporary tablespace	0	0.00	0.00
physical reads prefetch warmup	0	0.00	0.00
physical write IO requests	523	1.46	6.46
physical write bytes	5,783,552	16,140.61	71,401.88
physical write total IO requests	845	2.36	10.43
physical write total bytes	9,883,136	27,581.64	122,014.02
physical write total multi block requests	10	0.03	0.12
physical writes	706	1.97	8.72
physical writes direct	21	0.06	0.26
physical writes direct (lob)	5	0.01	0.06
physical writes direct temporary tablespace	0	0.00	0.00
physical writes from cache	685	1.91	8.46
physical writes non checkpoint	270	0.75	3.33

Physical IO disk bytes = physical read total bytes + physical write total bytes(误差可忽略)

值得注意的是这里 physical write total bytes 大致是 physical write bytes 的两倍。这应该是 physical write total bytes 统计的是磁盘的 IO，而这里，我们做了 ASM，normal redundancy，一份数据写了两遍的原因。

Load Profile 剩下的部分主要是关于各种执行情况的统计，除了 W/A MB processed 来自 V$PGASTAT（单位其实也是 Byte，不是 MB），其它数据都是来自于 V$SYSSTAT。

- Blocks Changes: 'db block changes'
- User calls: 'user calls'
- Parses: 'parse count (total)'
- Hard parses: 'parse count (hard)'
- Logons: 'logons cumulative'
- Executes: 'execute count'

- Rollbacks: 'user rollbacks'
- Tranasactions: 'user rollbacks' + 'user commits'
- W/A MB processed: 'bytes processed'

一般而言，Hard parses < Parses < Executes < User Calls。

AWR 的一般性介绍我想差不多就这些了，其它部分的介绍借助于一些更具体的 AWR 报告进行分析可能会更加方便和清晰。

5 AWR 报告分析 – 实战 1

构建 DSS 系统的第一步离不开数据加载，通过文本文件加载是最常见的方式，Oracle 提供了外部表加载的方法，即把一个文本文件当成一个正常的表来进行操作，通过类似 insert /*+ append */ into table select from external_table 的方式进行加载。

数据加载是一个 CPU-Bound 的过程，不过是通过什么工具，external table 也好，sqlldr 也好，imp 也好，impdp 也好。换句话说，如果连数据加载都出现 IO 瓶颈，这个系统的配置就说不过去了。

这个过程的 AWR 报告会是什么样子的呢？

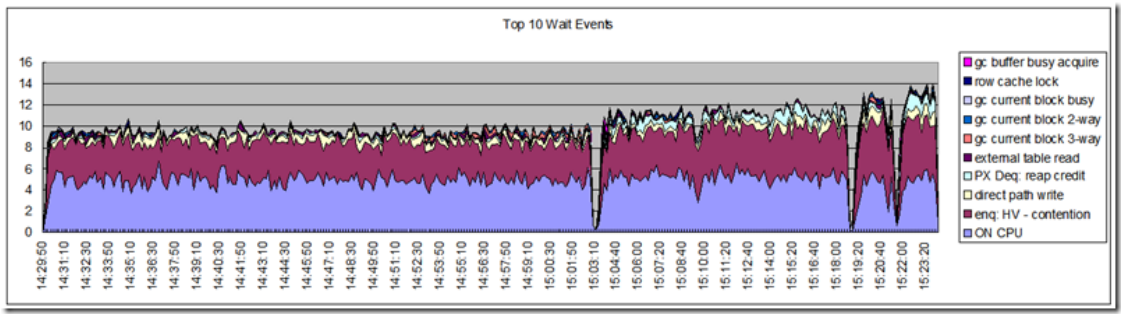
先做个一般的假定，从外部表加载数据到一个本地分区表。

Top 5 Timed Events 类似下面：

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		17,017		49.76	
enq: HV - contention	816,999	11,953	15	34.95	Other
direct path write	687,684	1,785	3	5.22	User I/O
PX Deq: reap credit	467,210,866	1,586	0	4.64	Other
external table read	45,430	479	11	1.40	User I/O

如果去抓取这段时间 DBA_HIST_ACTIVE_SESS_HISTORY 的数据，并转换为图表的话，我们会得到更形象的 Top 10 Wait Events。（如何实现这一步可以参考用 Oracle 实现 ASH 的数据透视图）



enq: HV – contention 是什么东西呢？

在 11.2 以前，对于分区表的 parallel direct-path load，Oracle 采用的是 brokered load 的方式，即所有的 PX Slaves 共享对每个分区的 high water mark 的访问，通过轮流持有 high water mark 实现对每个 segment 添加新的

blocks。这种方法对于充分利用 extent 的空间是有帮助的，不过带来的问题就是对 high water mark 的竞争，也就是这里的 enq: HV – contention。在执行计划中，这以 RANDOM LOCAL 标记。下面是一个例子：

```
-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name     | Rows  | Bytes | Cost (%CPU)| Time     |    TQ  |IN-OUT | PQ Distrib |
-------------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |          |  8168 |    14M|     2   (0)| 00:00:01 |        |       |            |
|   1 |  PX COORDINATOR          |          |       |       |            |          |        |       |            |
|   2 |   PX SEND QC (RANDOM)    | :TQ10001 |  8168 |    14M|     2   (0)| 00:00:01 | Q1,01  | P-&gt;S| QC (RAND)  |
|   3 |    LOAD AS SELECT        | TAB      |       |       |            |          | Q1,01  | PCWP  |            |
|   4 |     PX RECEIVE           |          |  8168 |    14M|     2   (0)| 00:00:01 | Q1,01  | PCWP  |            |
|   5 |      PX SEND RANDOM LOCAL| :TQ10000 |  8168 |    14M|     2   (0)| 00:00:01 | Q1,00  | P-&gt;P| RANDOM LOCA|
|   6 |       PX BLOCK ITERATOR  |          |  8168 |    14M|     2   (0)| 00:00:01 | Q1,00  | PCWC  |            |
|   7 |        EXTERNAL TABLE ACCESS FULL| ET_TAB   |  8168 |    14M|     2   (0)| 00:00:01 | Q1,00  | PCWP  |            |
-------------------------------------------------------------------------------------------------------------------
```

一个好消息是，11.2 引入了一种新的方式，叫做 PKEY distribution。在这种方式下，一个特定的分区只交给一个或多个特定的 PX slave 负责，这种方式不仅减少了对 high water mark 的争用，而且可以实现 Partition 内更好的压缩率。

6 AWR 报告分析 – 实战 2

有一次跟一个 QQ 上的朋友一起探讨了另一个对系统 CPU 进行度量的指标： CPU used by this session。

他刚好有一份 AWR 报告，在这份报告里，出现了严重的 CPU used by this session 和 DB CPU 不一致的现象。

下面是这份报告的一些片断：

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	10,517.75	90.06
DB CPU	8,934.22	76.50
parse time elapsed	626.47	5.36
connection management call elapsed time	291.54	2.50
hard parse elapsed time	162.35	1.39
PL/SQL execution elapsed time	34.34	0.29
failed parse elapsed time	17.57	0.15
sequence load elapsed time	2.82	0.02
hard parse (sharing criteria) elapsed time	0.26	0.00
repeated bind elapsed time	0.11	0.00
PL/SQL compilation elapsed time	0.10	0.00
DB time	11,677.97	
background elapsed time	389.86	
background cpu time	207.85	

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	5472	30-Nov-09 15:00:13	498	6.7
End Snap:	5473	30-Nov-09 16:00:16	498	6.6
Elapsed:		60.04 (mins)		
DB Time:		194.63 (mins)		

Statistic	Total
NUM_LCPUS	0
NUM_VCPUS	0
AVG_BUSY_TIME	90,933
AVG_IDLE_TIME	269,102
AVG_IOWAIT_TIME	3,986
AVG_SYS_TIME	26,456
AVG_USER_TIME	64,360
BUSY_TIME	1,821,080
IDLE_TIME	5,384,293
IOWAIT_TIME	81,942
SYS_TIME	531,445
USER_TIME	1,289,635
LOAD	0
OS_CPU_WAIT_TIME	2,003,700
RSRC_MGR_CPU_WAIT_TIME	0
PHYSICAL_MEMORY_BYTES	51,539,607,552
NUM_CPUS	20
NUM_CPU_CORES	10

Statistic	Total	per Second	per Trans
CPU used by this session	418,035	116.03	4.32

再做进一步的归纳：

OS Busy% = 1821080/(1821080+5384293) = 25%

Inst CPU% (using DB CPU) = 8934.22*100/ (1821080+5384293)=12%

Inst CPU% (using CPU used by this session) = 418035/ (1821080+5384293) = 6%

用 CPU used by this session 计算出的 CPU 利用率竟然只是用 DB CPU 计算出来的利用通率的一半！

我的第一个反应是在 Jonathan Lewis 网站看到的一篇相关文章，里面提到了 DB CPU 和 CPU used by this session 计算时的不同之处：

"Prior to 10g Oracle usually updated time figures at the end of each database call; but from 10g there are some views where time is updated more regularly.

The "*DB CPU*" from **v$sess_time_model** increases every six seconds, while the "*CPU used by this session*" from **v$sesstat** changes only at the end of the test."

如何验证这一点呢？

在浏览这份报告的 TOP SQL 时，我们发现了下面的现象：

Elapsed Time (s)	CPU Time (s)	Executions	Elap per Exec (s)	% Total DB Time	SQL Id
3,516	2,517	0		30.10	d9h33kacnjnqb
1,074	941	189	5.68	9.20	ajc56154r3zkq
696	433	5,582	0.12	5.96	63my0sfmjn4cz

这是从 SQL ordered by Elapsed Time 截取出来的 Top 3 SQL。TOP 1 的 SQL 用了 DB Time 的 30.10%，用了 2517s 的 CPU Time。但请注意它的 Executions 的值却为 0。也就是说，这里的 CPU Time 是还没有被计算入 CPU used by this session 这个指标里面的。

我们再把 2517s 加回来，看出误差缩小多少：(251700+418035)/ (1821080+5384293) = 9%，这时和用 DB CPU 计算出来的 12%还是有 1/4 的差距。

从这个例子可以看出，用 DB CPU 度量还是比用 CPU used by this session 来得准确的。特别在有大查询在跑的过程中抓的 AWR，这个误差很有可能会被放大。

一个有趣的实际例子。

7 总结

AWR 是分析数据库运行状况的利器，将其运用好可帮助 DBA 提早发现数据库中存在的问题并加以解决。文中主要对 DB CPU、DB Time、IO 等 AWR 报告中的数据进行了分析说明，当然分析 AWR 报告不能仅限于此，更需要 DBA 日积月累的经验。希望本文对想了解 AWR 的朋友有一定帮助。

Kaya 的个人简介

黄凯耀，英文名 Kaya

目前工作于 Oracle RealWorld Database Performance Group，一个隶属于 Oracle 公司总部数据库产品管理的核心团队。

大学及研究生时期专注于 Linux 应用开发和 Linux 内核开发工作，

2006 年加入 Oracle 公司至今，主要专注于
现实世界的数据库高性能，高可扩展性，高并发高压力的项目实践
Oracle Exadata Database Machine 上的客户真实项目的性能测试与优化
Oracle 客户所碰到的典型的性能问题的解决与方案建议
DSS 技术，如并行运算，数据分区，数据压缩，资源管理，ETL 等
OLTP 技术，如连接池管理，Cursor/Session 管理，Index 设计，应用架构优化等
SQL 性能优化，如 SQL 重写，执行计划分析，CBO 优化等
操作系统与数据库的资源监控与性能分析

个人站点：http://www.os2ora.com
Email： kaiyao.huang@gmail.com