## ORACLE性能诊断AWR报告分析

**报告分析结果示例：**

1. DB Time/Elapsed（DB Time除以CPU个数为每个CPU的DB Time）比值较小，服务器负荷较小
2. Load Profile：
   1. Sorts高达680，表明排序过多，需减少SQL代码中排序操作，有调整排序空间
   2. 硬解析比例较小，基本上皆为软解析，表明SQL重用率较高，符合预期
3. Instance Efficiency Percentages
   1. 缓冲区未等待率(buffer nowait %)、redo缓冲区未等待率(redo nowait %)、缓冲区命中率(buffer hit %)、内存排序率(in-memory sort %)、共享区命中率(library hit %)、软解析的百分比(soft parse %) 、闩锁命中率(latch hit %) 这几项指标皆为100%，符合预期
   2. execute to parse %)的值为负数，其计算公式为100 \* (1 - Parses/Executions)，说明shared pool 设置或者语句效率存在问题，造成反复解析，可通过调整session\_cached\_cursors解决
4. Shared Pool Statistics中Memory Usage %过小，说明共享池（share pool）内存存在较大的浪费，可适当调整
5. Top 5 Timed Events中log file sync排第一位，waits class为commit，可适当调整\_log\_io\_size
6. SQL Statistics部分中发现以下SQL消耗较大资源，可进行优化
   1. gbd19h4zdgbtj：SELECT COUNT(1) FROM SF\_DEG S WHERE S.ORGIN\_SHOP\_ID = :1 AND (S.DOC\_STATE = 65536)
7. 生成AWR报告

AWR由ORACLE自动产生，默认30分钟采集一次，保留5天的记录。但是也可以通过DBMS\_WORKLOAD\_REPOSITORY包来手工创建、删除和修改。使用脚本awrrpt.sql或awrrpti.sql来查看AWR报告，这两个脚本都在目录$ORACLE\_HOME/rdbms/admin中，报告可以保存为文本文件或HTML文件。

手动生成AWR报告的步骤如下：

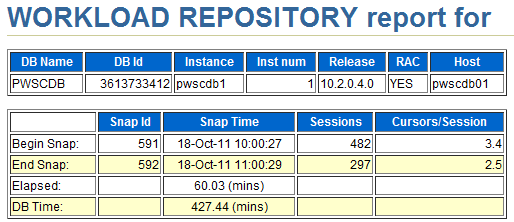
|  |
| --- |
| 1. su - oracle 切换至oracle用户 2. sqlplus / as sysdba 3. @?/rdbms/admin/awrrpt.sql 4. report\_type默认为HTML，可以不写 5. num\_days为距离当天的天数，昨天为2 6. begin\_snap为开始时间，填写时间前的编号 7. end\_snap为结束时间，填写时间前的编号 8. 输入文件名即可 |

通过脚本生成规定时长的AWR报告

|  |
| --- |
| 1. su - oracle 切换至oracle用户 2. cd /home/oracle/oracle\_monior 3. 执行./autoawr\_by\_time 90 4. 在report文件夹中查看生成的报告   以上命令表示生成时长为90分钟的AWR报告 |

1. AWR报告分析

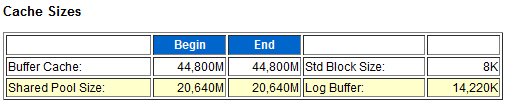
AWR报告头记录了数据库名称和起始快照时间，报告头中主要分析Elapsed（总时间）和DB Time（DB消耗的时间，不包括后台进行的消耗时间），如果DB Time/Elapsed比值较大，说明数据库系统压力较大，例如下图中系统包括16CPU(2\*8核)，每个cpu耗时26.7min，负载为26.7/60.03=44.5%，说明数据库服务器存在较大的负荷。



AWR报告总览包括了五个部分：缓存尺寸（Cache Sizes）、负载性能（Load Profile）、数据库效率（Instance Efficiency Percentages）、共享池统计（Shared Pool Statistics）、TOP5事件（Top 5 Timed Events）。这五个部分也就是整个报告核心，记录了数据库系统的关键性能参数和状况。

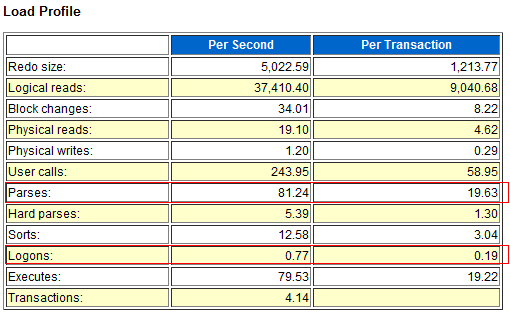
* **缓存尺寸（Cache Sizes）**

主要记录总的缓存大小Buffer Cache和SGA缓存尺寸Shared Pool Size，SGA是ORACLE中非常重要的内存共享区，对系统内的所有进程都是共享的。当多个用户同时连接到一个例程时，所有的用户进程、服务进程都可以共享使用这个SGA区。Shared pool可以分为库缓存（library cache）和数据字典缓存（dictionary cache）。Library cache存放了最近执行的SQL语句、存储过程、函数、解析树以及执行计划等。而dictionary cache则存放了在执行SQL语句过程中，所参照的数据字典的信息，包括SQL语句所涉及的表名、表的列、权限信息等。



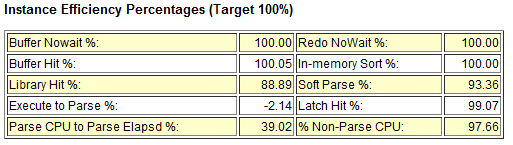
* **负载性能（Load Profile）**

这个部分记录了数据库负载情况，绝对值的分析意义不大，需要与之前的基线数据比较才具有更多的意义，单个的报告数据只说明应用的负载情况，绝大多数据并没有一个所谓“正确”的值。其中重要的几个对于Logons大于每秒1~2个，表明可能有争用问题；对于Hard parses大于每秒100，parses大于每秒300，表明硬解析太多，SQL重用率不高，需要解决SQL代码变量绑定问题，并调整共享池参数、调整cursor\_sharing参数；对于Sorts大于每秒100，表明排序过多，需要减少SQL代码中排序操作，或调整排序空间。



* **数据库效率（Instance Efficiency Percentages）**

记录了Oracle关键指标的内存命中率及数据库实例其它操作的效率，这个部分反应了数据库中最重要指标的命中率。

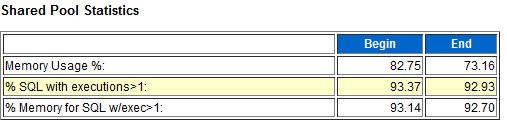


* 缓冲区未等待率(buffer nowait %)：指在缓冲区中获取buffer的未等待比率。该指标的值应接近100%，如果该值较低，则可能要增大buffer cache。
* redo缓冲区未等待率(redo nowait %)：指在redo缓冲区获取buffer的未等待比率。该指标的值应接近100%，如果该值较低，则有2种可能的情况：1）online redo log没有足够的空间；2）log切换速度较慢。
* 缓冲区命中率(buffer hit %)：指数据块在数据缓冲区中的命中率。该指标的值通常应在90%以上，否则，需要调整。如果持续小于90%，可能要加大db\_cache\_size。但有时，缓存命中率低并不意味着cache设置小了，可能是潜在的全表扫描降低了缓存命中率。
* 内存排序率(in-memory sort %)：指排序操作在内存中进行的比率。该指标的值应接近100%，如果指标的值较低，则表示出现了大量排序时的磁盘i/o操作，可考虑加大sort\_area\_size参数的值。
* 共享区命中率(library hit %)：该指标主要代表sql在共享区的命中率。该指标的值通常应在95%以上，否则需要考虑加大共享池（修改shared\_pool\_size参数值），绑定变量，修改cursor\_sharing等参数。
* 软解析的百分比(soft parse %)：该指标是指oracle对sql的解析过程中，软解析所占的百分比。该指标的值通常应在95%以上，如果低于80%，那么就可能sql基本没被重用，sql没有绑定变量，需要考虑绑定变量。
* 闩锁命中率(latch hit %)：指获得latch的次数与请求latch的次数的比率。该指标的值应接近100%，如果低于99%，需要分析闩锁竞争，明确是应用锁、数据字典锁、内存控制锁的哪一种。通过进一步分析Latch Statistics章节或动态性能视图v$session\_wait,v$latch,v$latch\_children。
* sql语句执行与解析的比率(execute to parse %)：指sql语句执行与解析的比率。该指标的值应尽可能到高，如果过低，可以考虑设置session\_cached\_cursors参数。

|  |
| --- |
| * Execute to Parse 指标反映了执行解析比 :其公式为 1-(parse/execute) , 目标为100% 及接近于只执行而不解析 * 在oracle中解析往往是执行的先提工作，但是通过游标共享 可以解析一次 执行多次， 执行解析可能分成多种场景： * hard coding => 硬编码代码 硬解析一次 ，执行一次， 则理论上其执行解析比 为 1:1 ，则理论上Execute to Parse =0 极差，且soft parse比例也为0% * 绑定变量但是仍软解析=》 软解析一次，执行一次 ， 这种情况虽然比前一种好 但是执行解析比(这里的parse，包含了软解析和硬解析)仍是1:1， 理论上Execute to Parse =0 极差， 但是soft parse比例可能很高 * 使用 静态SQL、动态绑定、session\_cached\_cursor、open cursors等技术实现的 解析一次，执行多次， 执行解析比为N:1， 则 Execute to Parse= 1- (1/N) 执行次数越多 Execute to Parse越接近100% ，这种是我们在OLTP环境中喜闻乐见的！ * 如果系统Parses > Executions，就可能出现该比率小于0的情况，该值<0通常说明shared pool设置或效率存在问题，造成反复解析。   通俗地说 soft parse反映了软解析率， 而软解析在oracle中仍是较昂贵的操作， 我们希望的是解析1次执行N次，如果每次执行均需要软解析，那么虽然soft parse%=100% 但是parse time仍可能是消耗DB TIME的大头。  Execute to Parse反映了 执行解析比，Execute to Parse和soft parse% 都很低 那么说明却是没有绑定变量 ， 而如果 soft parse% 接近99% 而Execute to Parse 不足90% 则说明没有执行解析比低， 需要通过 静态SQL、动态绑定、session\_cached\_cursor、open cursors等技术减少软解析。 |

* **共享池统计（Shared Pool Statistics）**

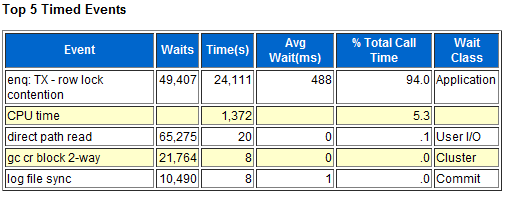
记录了在采集点时刻，共享池（share pool）内存被使用的比例。这个指标的值应保持在75%~90%,如果这个值太低，就浪费内存，如果太高，会使共享池外部的组件老化，如果sql语句被再次执行，则就会发生硬分析。其中执行次数大于1的sql比率（SQL with executions>1），如果此值太小，说明需要在应用中更多使用绑定变量，避免过多SQL解析。



* **TOP5事件（Top 5 Timed Events）**

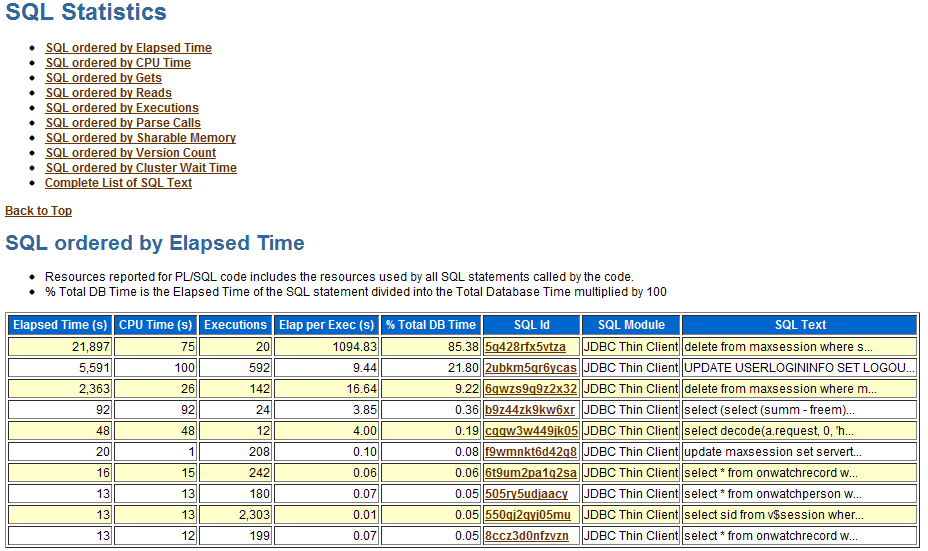
这个部分也是AWR报告中非常重要的部分，从这里可以看出等待时间在前五位的是什么事件，基本上就可以判断出性能瓶颈在什么地方。通常，在没有问题的数据库中，CPU time总是列在第一个，其他几类重要影响性能的事件分析如下。

* 缓冲区忙(buffer busy)：当一个会话想要访问缓存中的某个块，而这个块正在被其它会话使用时，将会产生该等待事件。这时候，其它会话可能正在从数据文件向缓存中的这个块写入信息，或正在对这个块进行修改。出现这个等待事件的频度不应大于1%。如果这个等待事件比较显著，则需要根据等待事件发生在缓存中的哪一块（如字段头部、回退段头部块、回退段非头部块、数据块、索引块等），采取相应的优化方法。
* 文件分散读取(db file scattered read)：该等待事件通常与全表扫描有关。因为全表扫描是被放入内存中进行的进行的，通常情况下它不可能被放入连续的缓冲区中，所以就散布在缓冲区的缓存中。如果这个等待事件比较显著，可能说明对于某些全表扫描的表，没有创建索引或没有创建合适的索引。尽管在特定条件下执行全表扫描可能比索引扫描更有效，但如果出现这种等待时，最好检查一下这些全表扫描是否必要。
* 文件顺序读取(db file sequential read)：该等待事件通常与单个数据块相关的读取操作有关。如果这个等待事件比较显著，可能表示在多表连接中，表的连接顺序存在问题，或者可能不合适地使用了索引。对于大量事务处理、调整良好的系统，这一数值大多是很正常的，但在某些情况下，它可能暗示着系统中存在问题。应检查索引扫描，以保证每个扫描都是必要的，并检查多表连接的连接顺序。另外db\_cache\_size?也是这些等待出现频率的决定因素。
* 队列(enqueue)：队列是一种保护共享资源的锁定机制。该锁定机制保护共享资源，如记录中的数据，以避免两个人在同一时间更新同一数据。如果enqueue等待事件比较显著，则需要根据enqueue等待类型，采取相应的优化方法。
* 闩锁释放(latch free)：latch是一种低级排队机制(它们被准确地称为相互排斥机制),用于保护系统全局区域(sga)中共享内存结构。该等待事件意味着进程正在等待其他进程已持有的latch。对于常见的latch等待通常的解决方法：1）share pool latch：在oltp应用中应该更多的使用绑定变量以减少该latch的等待。2）library cache latch：同样的需要通过优化sql语句使用绑定变量减少该latch的等待。
* 日志文件同步(log file sync)：这个等待事件是指当一个会话完成一个事务（提交或者回滚数据）时，必须等待lgwr进程将会话的redo信息从日志缓冲区写到日志文件后，才能继续执行下去。这个等待事件的时间过长，可能是因为commit太频繁或者lgwr进程一次写日志的时间太长（可能是因为一次log io size太大），可调整\_log\_io\_size。



* **SQL统计（SQL Statistics）**

AWR报告中还有一块对性能影响最大的指标，TOP SQL统计。本节按各种资源分别列出对资源消耗最严重的SQL语句，并显示它们所占统计期内全部资源的比例，提供给我们调优依据。



* **SQL ordered by Elapsed Time**： 记录了执行总和时间的SQL，记录的是监控范围内该SQL的执行时间总和，需要综合分析CPU时间（CPU Time）和执行次数（Executions）才能得到单个SQL的代价。单次执行开销较大的SQL属于重点优化之列。
* **SQL ordered by CPU Time**： 记录了执行占CPU时间总和时间最长的SQL，再CPU消耗较大的系统中，重点优化此类SQL。
* **SQL ordered by Gets**： 记录了执行占总buffer gets(逻辑IO)的SQL，查找总的缓冲区获取比较高的SQL，并根据平均每次执行缓冲区获取的数量判断优化的余地有多大。优化这些SQL，有助于减少CPU开销以及数据缓冲池相关的闩锁竞争。
* **SQL ordered by Reads**：记录了执行占总磁盘物理读(物理IO)的SQL，查找总的物理读比较高的SQL，并根据平均每次执行物理读的数量判断优化的余地有多大。优化这些SQL，有助于减少I/O开销和CPU开销。
* **SQL ordered by Executions**：记录了按照SQL的执行次数排序的SQL，执行次数多的SQL也是需要重点优化，使sql语句中的子操作执行次数尽量少。
* **SQL ordered by Parse Calls**：记录了解析次数排序的SQL，避免出现硬解析，采用使用绑定变量等方式。
* **SQL ordered by Sharable Memory**：记录了SQL占用library cache的大小的SQL。
* **SQL ordered by Version Count**：记录了SQL的打开子游标的SQL。
* **SQL ordered by Cluster Wait Time**：记录了集群的等待时间的SQL。

1. 其他：

**死锁的定位：**通过检查数据库表，能够检查出是哪一条语句被死锁，产生死锁的机器是哪一台。  
1）用dba用户执行以下语句

|  |
| --- |
| select username,lockwait,status,machine,program from v$session where sid in  (select session\_id from v$locked\_object) |

如果有输出的结果，则说明有死锁，且能看到死锁的机器是哪一台。字段说明：Username：死锁语句所用的数据库用户；

Lockwait：死锁的状态，如果有内容表示被死锁。

Status： 状态，active表示被死锁

Machine： 死锁语句所在的机器。

Program： 产生死锁的语句主要来自哪个应用程序。

2）用dba用户执行以下语句，可以查看到被死锁的语句。

|  |
| --- |
| select sql\_text from v$sql where hash\_value in  (select sql\_hash\_value from v$session where sid in  (select session\_id from v$locked\_object)) |

**死锁的解决方法：**一般情况下，只要将产生死锁的语句提交就可以了，但是在实际的执行过程中。用户可能不知道产生死锁的语句是哪一句。可以将程序关闭并重新启动就可以了。

　经常在Oracle的使用过程中碰到这个问题，所以也总结了一点解决方法。

1. 查找死锁的进程：

|  |
| --- |
| sqlplus / as sysdba  SELECT s.username,l.OBJECT\_ID,l.SESSION\_ID,s.SERIAL#,  l.ORACLE\_USERNAME,l.OS\_USER\_NAME,l.PROCESS  FROM V$LOCKED\_OBJECT l,V$SESSION S WHERE l.SESSION\_ID=S.SID; |

2）kill掉这个死锁的进程：

|  |
| --- |
| alter system kill session ‘sid,serial#’; （其中sid=l.session\_id） |

3）如果还不能解决：

|  |
| --- |
| select pro.spid from v$session ses,v$process pro where ses.sid=XX and ses.paddr=pro.addr;  #其中sid用死锁的sid替换  ps -ef|grep spid  #其中spid是这个进程的进程号，kill掉这个Oracle进程  select A.SQL\_TEXT, B.USERNAME, C.OBJECT\_ID, C.SESSION\_ID,         B.SERIAL#, C.ORACLE\_USERNAME,C.OS\_USER\_NAME,C.Process,         ''''||C.Session\_ID||','||B.SERIAL#||''''  from v$sql A, v$session B, v$locked\_object C  where A.HASH\_VALUE = B.SQL\_HASH\_VALUE and  B.SID = C.Session\_ID |