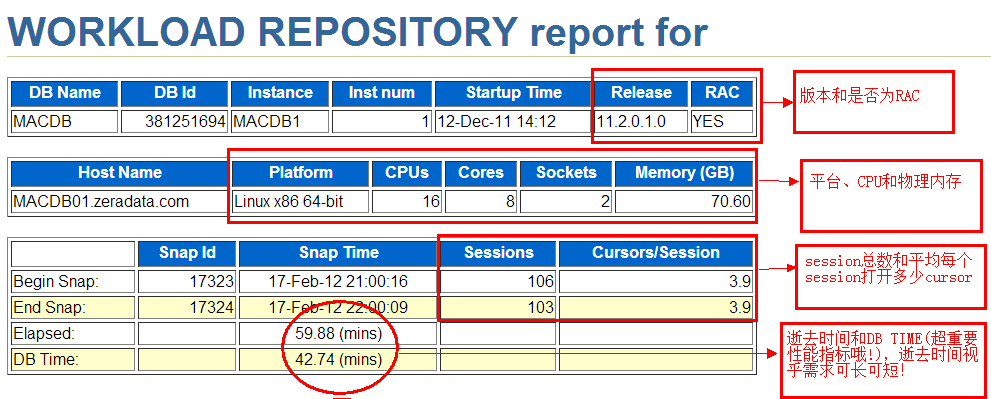
>db time



Cursors/session è open\_cursors ORA-100诊断

elapsed: 自然流逝的时间：通常60mins，可以自己配置

db time = 数据库上的负载 即 所有前台进程花费的总时间（包括cpu time，io time，所有非空闲等待，不要忘记cpu on

queue time）-反应用户工作负载，所有没有后台进程

response time：响应时间 不同于db time

DB Time描绘了数据库总体负载，但要和elapsed time逝去时间结合其他来。

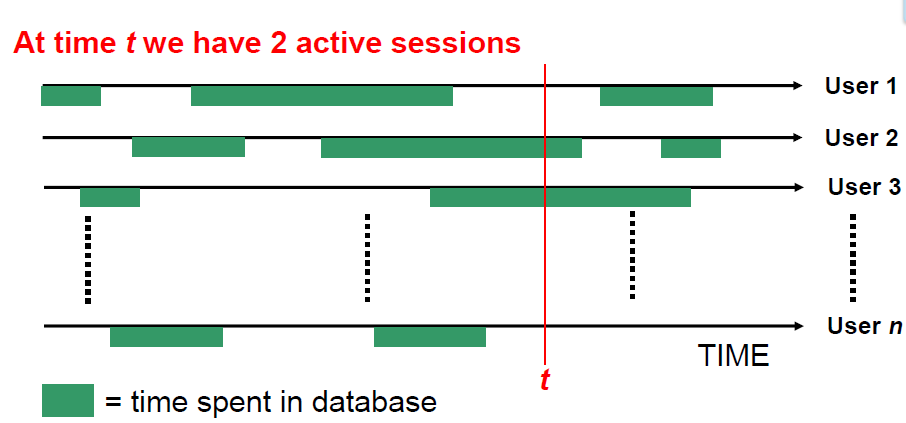
Average Active Session AAS= DB time/Elapsed Time DB Time ，

Elapsed Time =60 min AAS=60/60=1 负载一般 DB Time= 1min ,

Elapsed Time= 60 min AAS= 1/60 负载很轻 DB Time= 60000 min，

Elapsed Time= 60 min AAS=1000 è 系统hang了吧？

AAS = time spent in database（using cpu，cpu on queue time。。。）



**AAS =DB TIME/ELAPSED**

**DB TIME= DB CPU + Non-Idle Wait + Wait on CPU queue**

如果仅有2个逻辑CPU，而2个session在60分钟都没等待事件，一直跑在CPU上，那么：

DB CPU= 2 \* 60 mins ， DB Time = 2\* 60 + 0 + 0 =120

AAS = 120/60=2 正好等于OS load 2。

如果有3个session都100%仅消耗CPU，那么总有一个要wait on queue

DB CPU = 2\* 60 mins ，wait on CPU queue= 60 mins

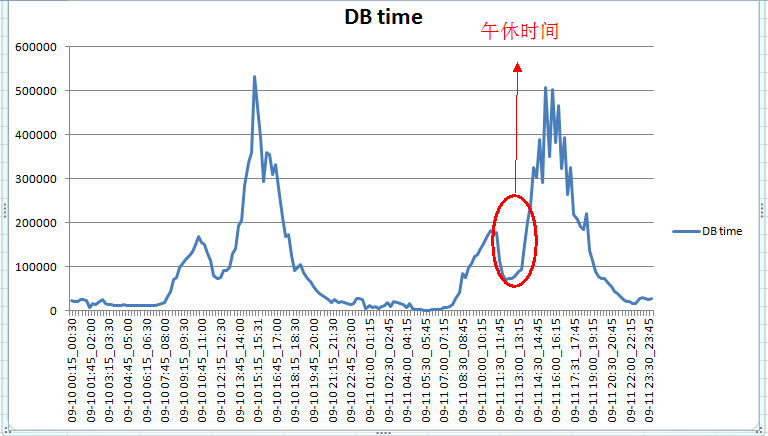
AAS= (120+ 60)/60=3 主机load 亦为3，此时vmstat 看waiting for run time

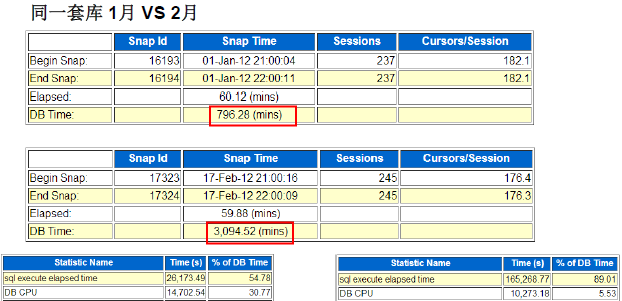
真实世界中？

DB Cpu = xx mins ，

Non-Idle Wait= enq:TX + cursor pin S on X + latch : xxx + db file sequential read + ………..

对比所有dbtime 可以做一个负载折线图





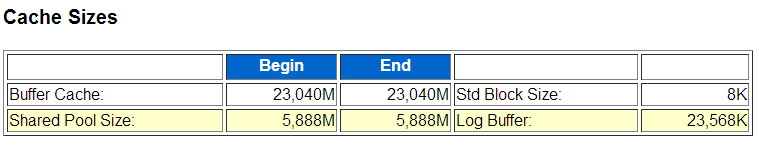
以上可以看到db time出现明显的差异，而db cpu相差不多，说明什么？Non-Idle Wait + Wait on CPU queue增加！！

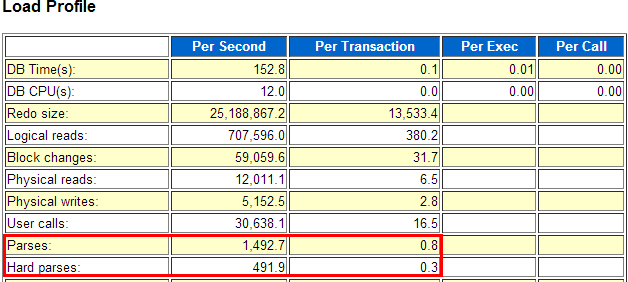
>兵马未动，cache先行

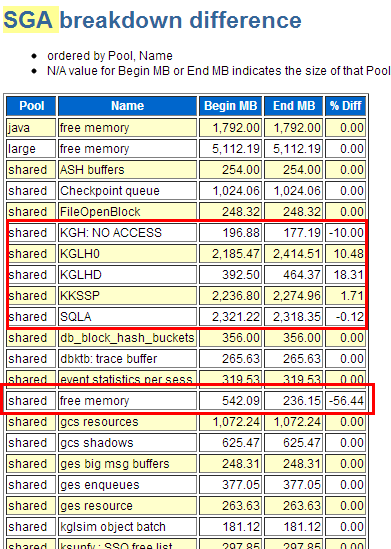
内存管理方式：MSMM、ASMM(sga\_target)、AMM(memory\_target)

小内存有小内存的问题， 大内存有大内存的麻烦！ ORA-04031???!!

**Buffer cache**和**shared pool size** 的 **begin/end**值在**ASMM**、 **AMM**和 **11gR2 MSMM**下可是会动的哦



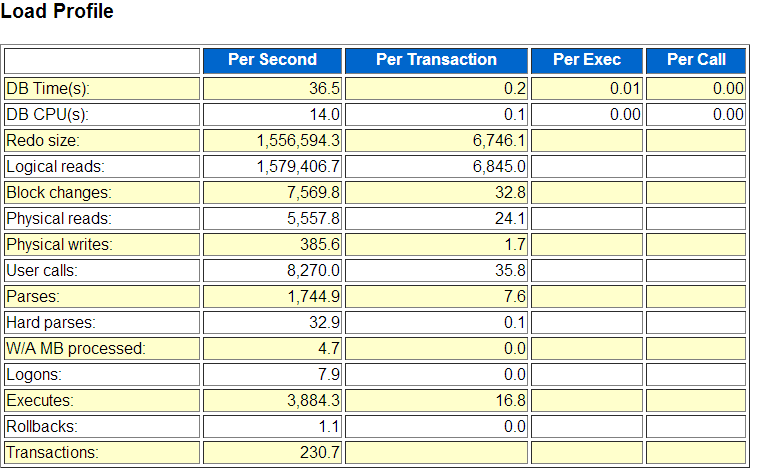




注意%diff： 直接看变化率

>LOAD PROFILE

通常需要结合baseline分析



信息量很大

Redo size 单位 bytes，redo size可以用来估量update/insert/delete的频率，大的redo size往往对lgwr写日志，和arch归档造成I/O压力， Per Transaction可以用来分辨是 大量小事务， 还是少量大事务 如上例每秒redo 约1.5MB ，每个事务6k，符合OLTP特征

Logical Read单位 次数\*块数， 相当于 “人\*次”， 如上例 1579406 \* db\_block\_size=12GB/s ， 逻辑读耗CPU，主频和CPU核数都很重要，逻辑读高则DB CPU往往高，也往往可以看到latch: cache buffer chains等待。 大量OLTP系统(例如siebel)可以高达几十乃至上百Gbytes。

Block changes 单位 次数\*块数 ， 描绘数据变化频率

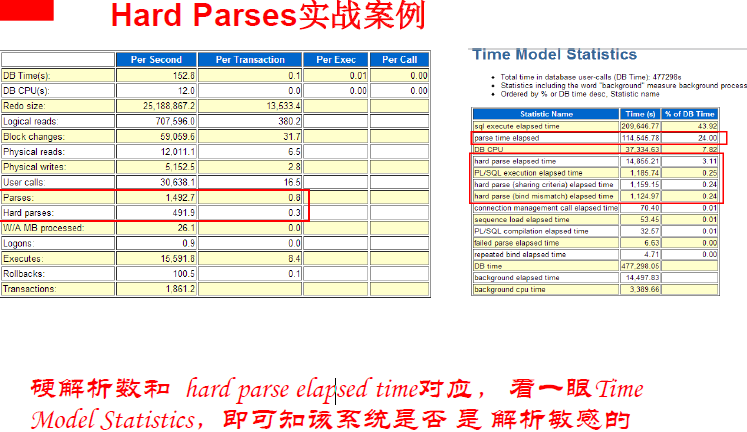
Physical Read 单位次数\*块数， 如上例 5557 \* 8k = 43MB/s， 物理读消耗IO读，体现在IOPS和吞吐量等不同纬度上；但减少物理读可能意味着消耗更多CPU。好的存储 每秒物理读能力达到几GB，例如Exadata。

Physical writes单位 次数\*块数，主要是DBWR写datafile，也有direct path write。 dbwr长期写出慢会导致定期log file switch(checkpoint no complete) 检查点无法完成的前台等待。

User Calls 单位次数，用户调用数，more details from internal

Parses，解析次数，包括软解析+硬解析，软解析优化得不好，则夸张地说几乎等于每秒SQL执行次数。 即执行解析比1:1，而我们希望的是 解析一次到处运行哦！

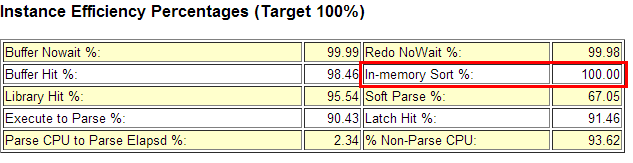
Hard Parses ：万恶之源． Cursor pin s on X， library cache: mutex X ， latch: row cache objects /shared pool……………..。 硬解析最好少于每秒20次

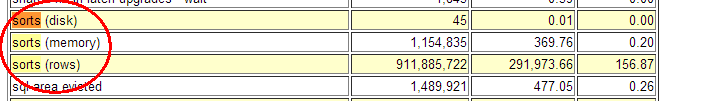


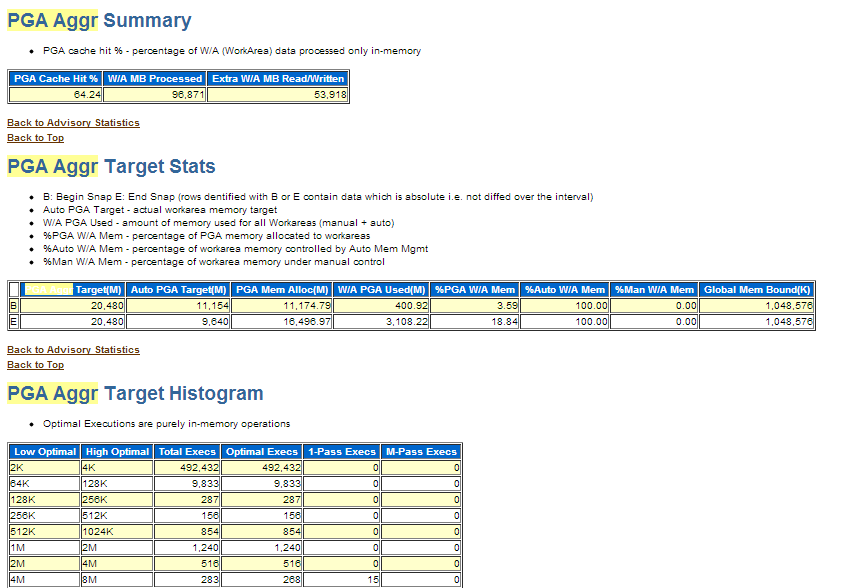
W/A MB processed

**W/A MB processed :** 单位**MB W/A workarea workarea** 中处理的数据数量

结合 **In-memory Sort%**， **sorts (disk)， PGA Aggr**一起看

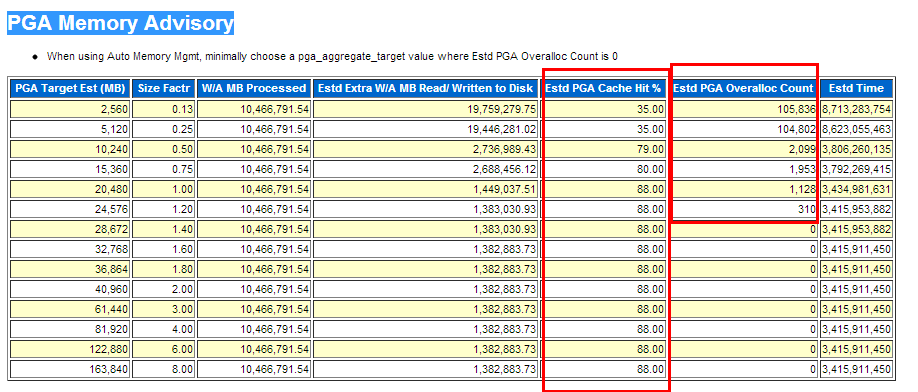






pga memory advisory

pga\_aggregate\_target过小会导致PGA overalloc过载， 但对于变态的HASH/SORT需求，再大的PGA也达不到cache hit 100%



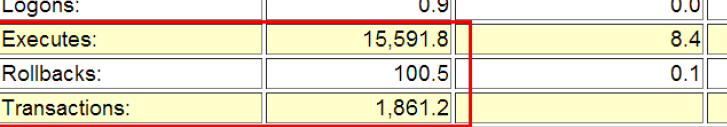
Logons 登陆次数， logon storm 登陆风暴，结合AUDIT审计数据一起看。短连接的附带效应是游标缓存无用,以下为短连接变长连接后的优化效果



Executes 执行次数，反应执行频率

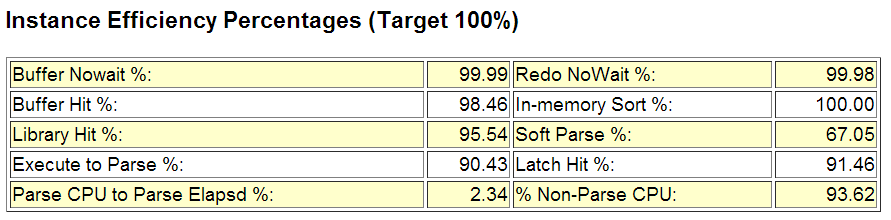
Rollback 回滚次数， 反应回滚频率， 但是这个指标不太精确，参考而已，别太当真

Transactions 每秒事务数，和数据库层的TPS，可以看做压力测试或比对性能是的一个指标，孤立看无意义（比之前每秒执行的事物多）。



**>Instance Efficiency Percentages (Target 100%)**

基于命中率的调优方法论已经过时，但仍具有参考价值



The section provides information about hit ratios for different memory components. These ratios, tells how often a particular data is found in a respecitive memory structure. Details about hit ratios are as mentioned delow,

**Buffer Hit:** Shows % of times particular block was found in buffer cache, instead of reading it from disk.

**Buffer Nowait:** Indicates % of times data buffers were accessed directly without any wait time.

**Library Hit%:** Shows % of times SQL and PL/SQL found in shared pool.

**In-Memory Sort %:** Shows % of times Sorting operations happened in memory than in the disk (temporary tablespace).

**Soft parse %:** Shows % of times the SQL in shared pool is used.

**Latch Hit %:** Shows % of time latches are acquired without having to wait.

As per Oracle AWR report the target should be 100% for these ratios. But in reality this is not possible always. Hence the ratio above 80% is always healthy.

全部是越高越好！

Buffer nowait%: session申请一个buffer(兼容模式)不等待的次数比例。

Buffer HIT%: 经典的经典，高速缓存命中率，反应物理读和缓存命中间的纠结，但这个指标即便99% 也不能说明物理读等待少了

Redo nowait%: session在生成redo entry时不用等待的比例，redo相关的资源争用例如redo space request争用可能造成生成redo时需求等待。此项数据来源于v$sysstat中的redo log space requests/redo entries（redo先生成在pga中，在写入log buffer）

In-memory Sort%:这个指标因为它不计算workarea中所有的操作类型，所以现在越来越鸡肋了。 纯粹在内存中完成的排序比例。数据来源于v$sysstat statistics sorts (disk) 和 sorts (memory).

Library Hit%: library cache命中率，申请一个library cache object例如一个SQL cursor时，其已经在library cache中的比例。 数据来源 V$librarycache的pins和pinhits。 合理值：>95%

Soft Parse: 软解析比例，无需多说的经典指标，数据来源v$sysstat statistics的parse count(total)和parse count(hard)。 合理值>95%

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。 数据来源v$sysstat statistics parse count (total) 和execute count

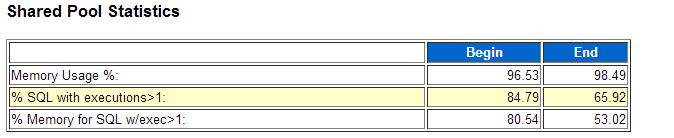
Latch Hit%: willing-to-wait latch闩申请不要等待的比例。 数据来源V$latch gets和misses

Parse CPU To Parse Elapsd:该指标反映了 快照内解析CPU时间和总的解析时间的比值(Parse CPU Time/ Parse Elapsed Time)； 若该指标水平很低，那么说明在整个解析过程中 实际在CPU上运算的时间是很短的，而主要的解析时间都耗费在各种其他非空闲的等待事件上了(如latch:shared pool,row cache lock之类等) 数据来源 V$sysstat 的 parse time cpu和parse time elapsed

%Non-Parse CPU 非解析cpu比例，公式为 (DB CPU – Parse CPU)/DB CPU， 若大多数CPU都用在解析上了，则可能好钢没用在刃上了。 数据来源 v$sysstat 的 parse time cpu和 cpu used by this session

**>Shared Pool Statistics**

反应**SQL**重用率和共享池中 **cursor**对内存的使用



**% SQL with executions>1:** Shows % of SQLs executed more than 1 time. The % should be very near to value 100.

**% memory for SQL w/exec>1:** From the memory space allocated to cursors, shows which % has been used by cursors more than 1.

The ratio above 80% is always healthy.

该环节提供一个大致的SQL重用及shared pool内存使用的评估。 应用是否共享SQL? 有多少内存是给只运行一次的SQL占掉的，对比共享SQL呢？

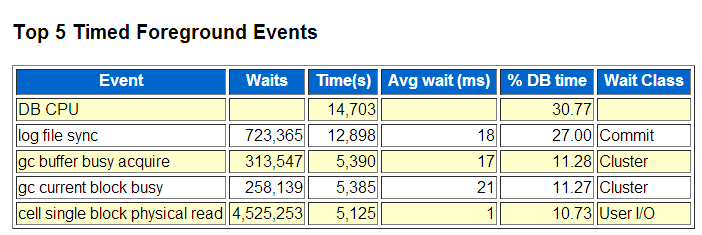
如果该环节中% SQL with executions>1的 比例 小于%90 ， 考虑用下面链接的SQL去抓 硬编码的非绑定变量SQL语句。

利用FORCE\_MATCHING\_SIGNATURE捕获非绑定变量SQL

<http://www.askmaclean.com/archives/%E5%88%A9%E7%94%A8force_matching_signature%E6%8D%95%E8%8E%B7%E9%9D%9E%E7%BB%91%E5%AE%9A%E5%8F%98%E9%87%8Fsql.html>

**>Top 5 Timed Foreground Events DB CPU/Cpu Time**

**Top 5** 万众瞩目，**DBA**为你倾倒！



### **db file scattered read:**

This event indicates wait due to full table scans or index fast full scans.

To avoid this event, identify all the tables on which FTS is happening and create proper indexes so that oracle will do Index scans instead of FTS. The index scan will help in reducing no. of IO operations.

To get an idea about tables on which FTS is happening please refer to “Segment Statistics” -> “Segments By Physical Read” section of AWR report. This section lists down both Tables and Indexes on which Physical Reads are happening. Please note that physical reads doesn’t necessarily means FTS but a possibility of FTS.

### **db file sequential read:**

The event indicates that index scan is happening while reading data from table. High no. of such event may be a cause of unselective indexes i.e. oracle optimizer is not selecting proper indexes from set of available indexes. This will result in extra IO activity and will contribute to delay in SQL execution.

Generally high no. is possible for properly tuned application having high transaction activity.

### **buffer buzy wait:**

Indicates that particular block is being used by more than one processes at the same. When first process is reading the block the other processes goes in a wait as the block is in unshared more. Typical scenario for this event to occur is, when we have batch process which is continuously polling database by executing particular SQL repeatedly and there are more than one parallel instances running for the process. All the instances of the process will try to access same memory blocks as the SQL they are executing is the same. This is one of the situation in which we experience this event.

### **buffer buzy wait:**

Indicates that particular block is being used by more than one processes at the same. When first process is reading the block the other processes goes in a wait as the block is in unshared more. Typical scenario for this event to occur is, when we have batch process which is continuously polling database by executing particular SQL repeatedly and there are more than one parallel instances running for the process. All the instances of the process will try to access same memory blocks as the SQL they are executing is the same. This is one of the situation in which we experience this event.

### **enq: TX - row lock contention:**

Oracle maintence data consistency with the help of locking mechanism. When a particular row is being modified by the process, either through Update/ Delete or Insert operation, oracle tries to acquire lock on that row. Only when the process has acquired lock the process can modify the row otherwise the process waits for the lock. This wait situation triggers this event. The lock is released whenever a COMMIT is issued by the process which has acquired lock for the row. Once the lock is released, processes waiting on this event can acquire lock on the row and perform DML operation.

基于Wait Interface的调优是目前的主流！每个指标都重要！

基于命中比例的调优，好比是统计局的报告， 张财主家财产100万，李木匠家财产1万， 平均财产50.5万。

基于等待事件的调优，好比马路上100辆汽车的行驶记录表，上车用了几分钟， 红灯等了几分钟，拥堵塞了几分钟。。。

Mysql梦寐以求的东西……

CPU 上在干什么？

逻辑读？ 解析？Latch spin? PL/SQL、函数运算?

DB CPU/CPU time是Top 1 是好事情吗？ 未必！

注意DB CPU不包含 wait on cpu queue！

**结合Host CPU、Instance CPU、 SQL ordered by CPU Time,Operating System Statistics一起看哦！**

**db file sequential read- Top event**

**Avg wait time应当小于20ms**

**”db file sequential read”单块读等待是一种最为常见的物理IO等待事件，这里的sequential指的是将数据块读入到相连的内存空间中(contiguous memory space)，而不是指所读取的数据块是连续的。该wait event可能在以下情景中发生:**

**<http://www.askmaclean.com/archives/db-file-sequential-read-wait-event.html>**

IMG_274

db file scattered read - Top event

Avg wait time应当小于20ms

常见原因 Fast Full scan Index ， FULL SCAN large table

IMG_275

IMG_276

**db file sequential/scattered read - Top event 参考SQL ordered by Reads，Segments by Physical Reads**

**Log file sync**蝴蝶效应

Log file sync ==》 enq: TX ，gc buffer busy，buffer busy wait

等待事件的混沌理论，性能不是线性的，而是多纬度的

log file parallel write慢=> log file sync慢=>commit慢，commit慢则释放行锁慢。 Rac flush redo也受到写redo慢的影响，则出现gc buffer busy release/acquire，前后相互作用è enq:TX 大幅出现

Enq:TX row lock出现在哪里？哪些语句受到GC buffer busy影响？

最主要是update和insert 受影响，前台处理业务速度放慢。方向对了就处处对得上了

Global Buffer Busy =》gc buffer busy acquire/release 受影响的segment

directory path read

直接路径读：不读到buffer cache，但是会加大io。

oracle11g自己判断是否使用directory path read，如果表的数据量是bufferchache的2倍则使用directory path read。

性能优化的多维度理论

•增加了cpuè更大的并发量，更多的并发争用

•调整了Io存储è 更少的IO，更多的CPU计算，更高的cpu使用率

•Redo写得慢è 影响commit，造成enq:tx和gc buffer busy等待等

•Datafile写得慢è 检查点完不成，日志无法切换，前台DML hang

•Sequence nocacheè INSERT index很容易造成enq:index contention，和row cache lock和 enq:SQ

•通过数据库手段优化了性能è 应用本身设计的瓶颈越来越凸显

不给应用开大手术，纯数据库优化的极限

[www.askmaclean.com](http://www.askmaclean.com)

tuning

or

<http://www.askmaclean.com/archives/tag/tuning>