开Oracle调优鹰眼，深入理解AWR性能报告

ORA-ALLSTARS Exadata用户组

(千人群) 行业交流-IT/互联网

刘相兵(Maclean Liu)
liu.maclean@gmail.com

ORA-ALLSTARS Exadata用户组QQ群:23549328

About Me

ORACLE
Certified Master
Oracle Database 10g
Administrator

ORACLE
Certified Master
Oracle Database 11g
Administrator

- **Email:liu.maclean@gmail.com**
- **Blog:www.askmaclean.com**
- **Oracle Certified Database Administrator Master 10g and 11g**
- **Over 7 years experience with Oracle DBA technology**
- **Over 8 years experience with Linux technology**
- **Member Independent Oracle Users Group**
- **Member All China Users Group**
- **Presents for advanced Oracle topics: RAC, DataGuard, Performance Tuning and Oracle Internal.**

IOUG
independent oracle users group
For the Complete Technology & Database Professional

ACOUG
All China Oracle User Group
MEMBER

Hawk Eyes 看AWR的鹰眼= 基础理论夯实+看过500份以上AWR

啥是AWR?

AWR (Automatic Workload Repository)

一堆历史性能数据，放在SYSAUX表空间上， AWR和SYSAUX都是10g出现的，是Oracle调优的关键特性； 大约1999年左右开始开发，已经有15年历史

默认快照间隔1小时，10g保存7天、11g保存8天; 可以通过DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS修改

DBA_HIST_WR_CONTROL

AWR程序核心是dbms_workload_repository包

@?/rdbms/admin/awrrpt 本实例

@?/rdbms/admin/awrrpti RAC中选择实例号

AWR数据从哪里来，放哪里去？

基础指标统计：

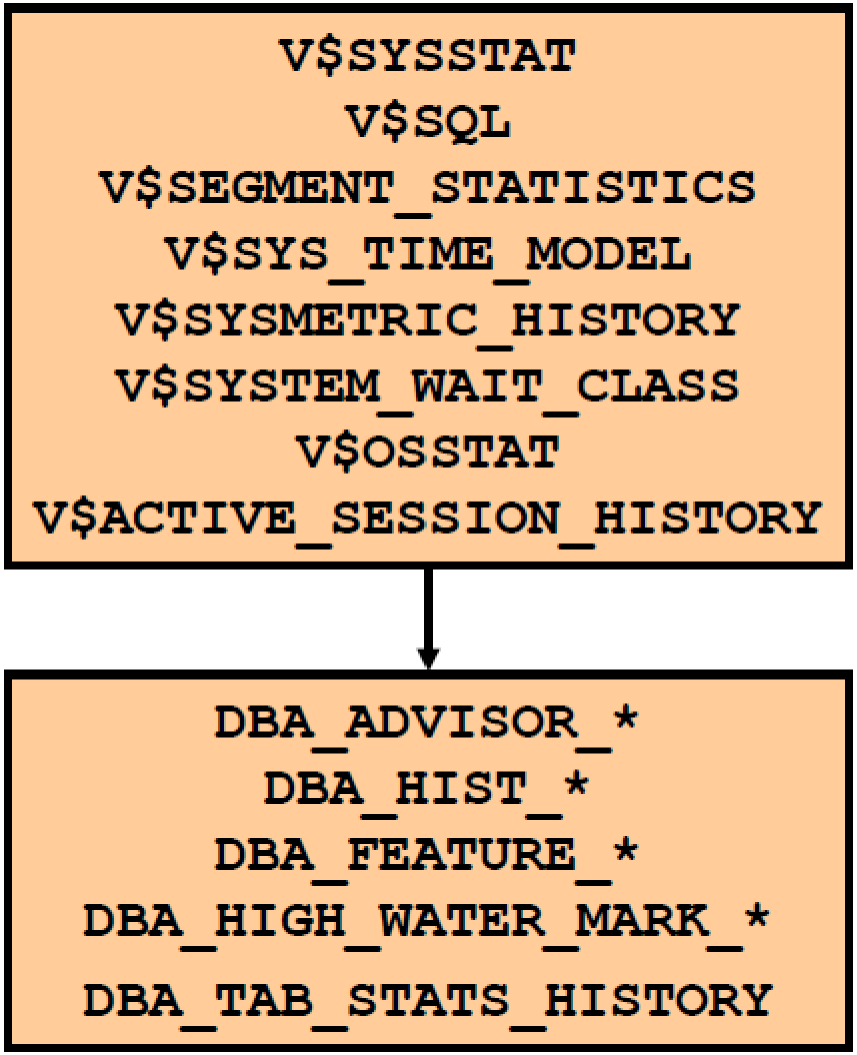
◆SQL和优化器指标

◆OS指标

◆等待事件类型

◆时间指标

- 度量
- **ASH Active Session History**
- 建议器**advisor**
- 快照指标
- 数据库特性使用情况



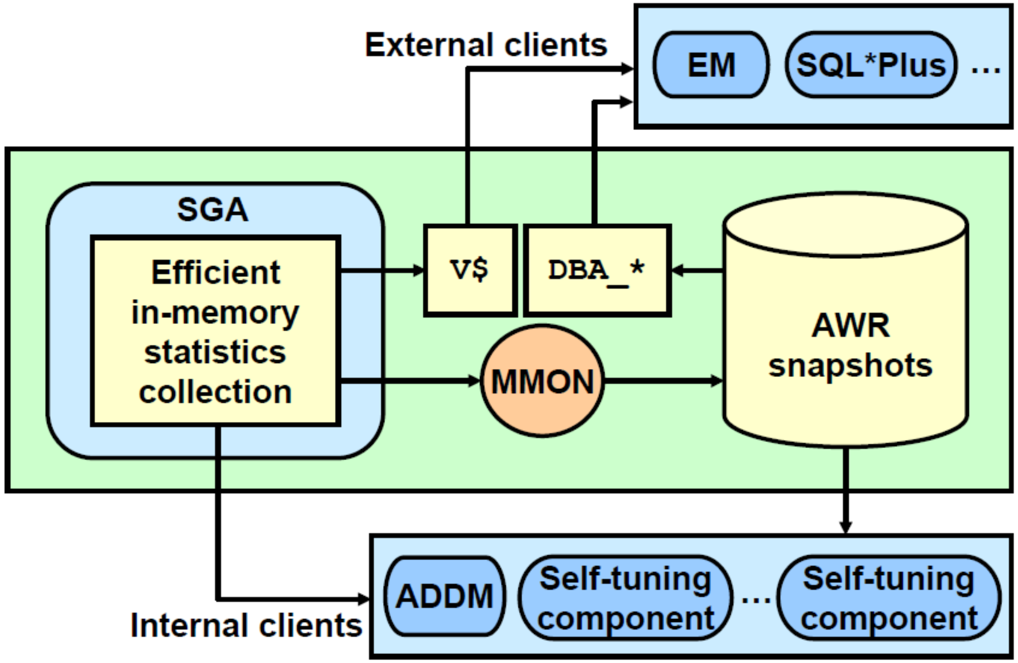
V$SYSSTAT
V$SQL
V$SEGMENT_STATISTICS
V$SYS_TIME_MODEL
V$SYSMETRIC_HISTORY
V$SYSTEM_WAIT_CLASS
V$OSSTAT
V$ACTIVE_SESSION_HISTORY

DBA_ADVISOR_*
DBA_HIST_*
DBA_FEATURE_*
DBA_HIGH_WATER_MARK_*
DBA_TAB_STATS_HISTORY

谁维护AWR?

主要是MMON(Manageability Monitor Process)和它的小工进程(m00x)
MMON的功能包括:

1. **启动slave进程m00x去做AWR快照**
2. **当某个度量阀值被超过时发出alert告警**
3. **为最近改变过的SQL对象捕获指标信息**



mmon若hang则意味着AWR不可用!!

AWR小技巧

手动执行一个快照：

Exec dbms_workload_repository.create_snapshot; (这个要背出来哦，用的时候去翻手册，丢脸哦 ☺**!)**

创建一个**AWR**基线

Exec DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE(start_snap_id，end_snap_id ,baseline_name);

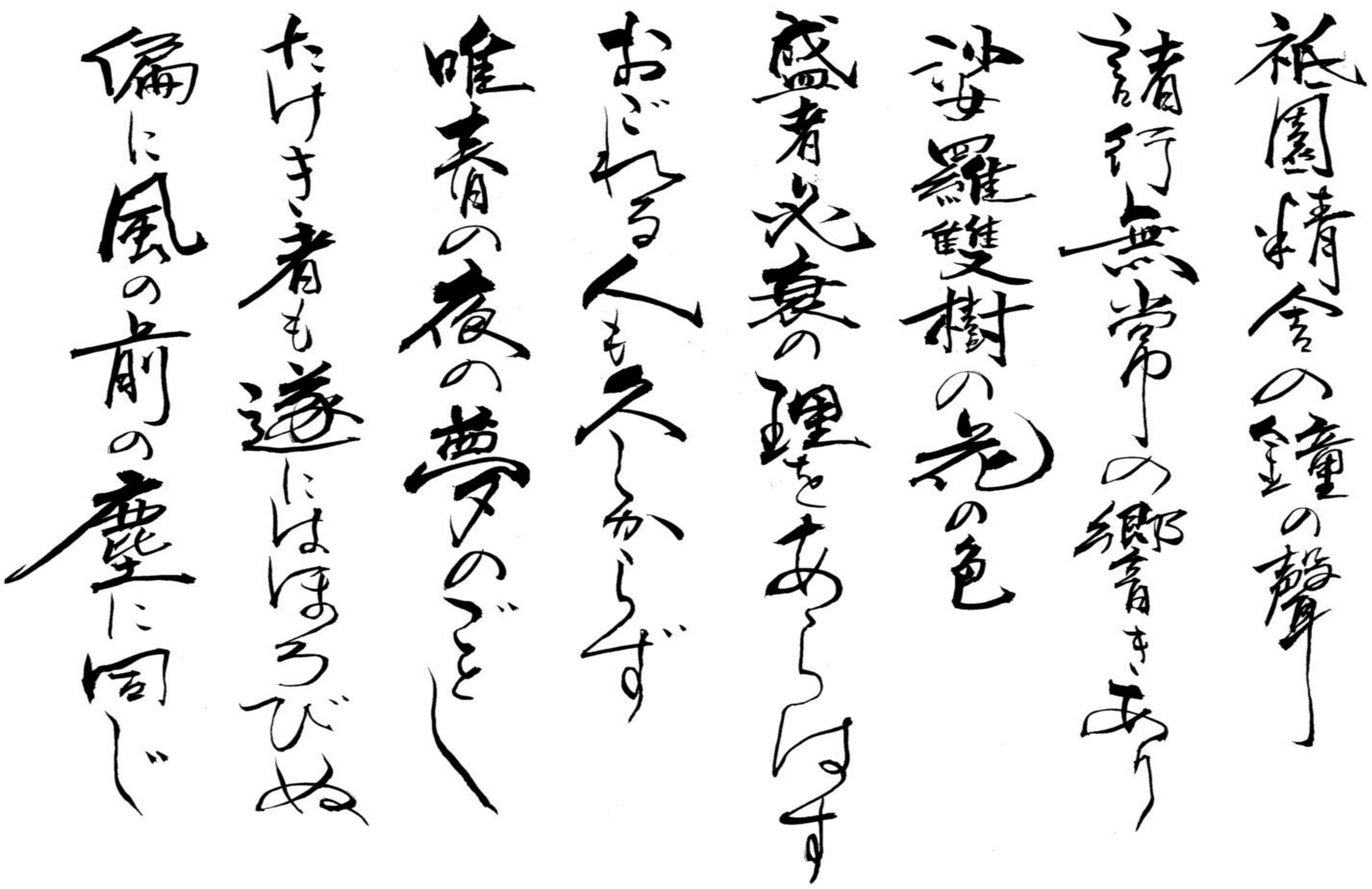
@?/rdbms/admin/awrddrpt AWR比对报告

@?/rdbms/admin/awrgrpt RAC 全局**AWR**

自动生成**AWR HTML**报告：

http://www.oracle-base.com/dba/10g/generate_multiple_awr_reports.sql

AWR开篇 卷首灌顶，惊鸿一瞥

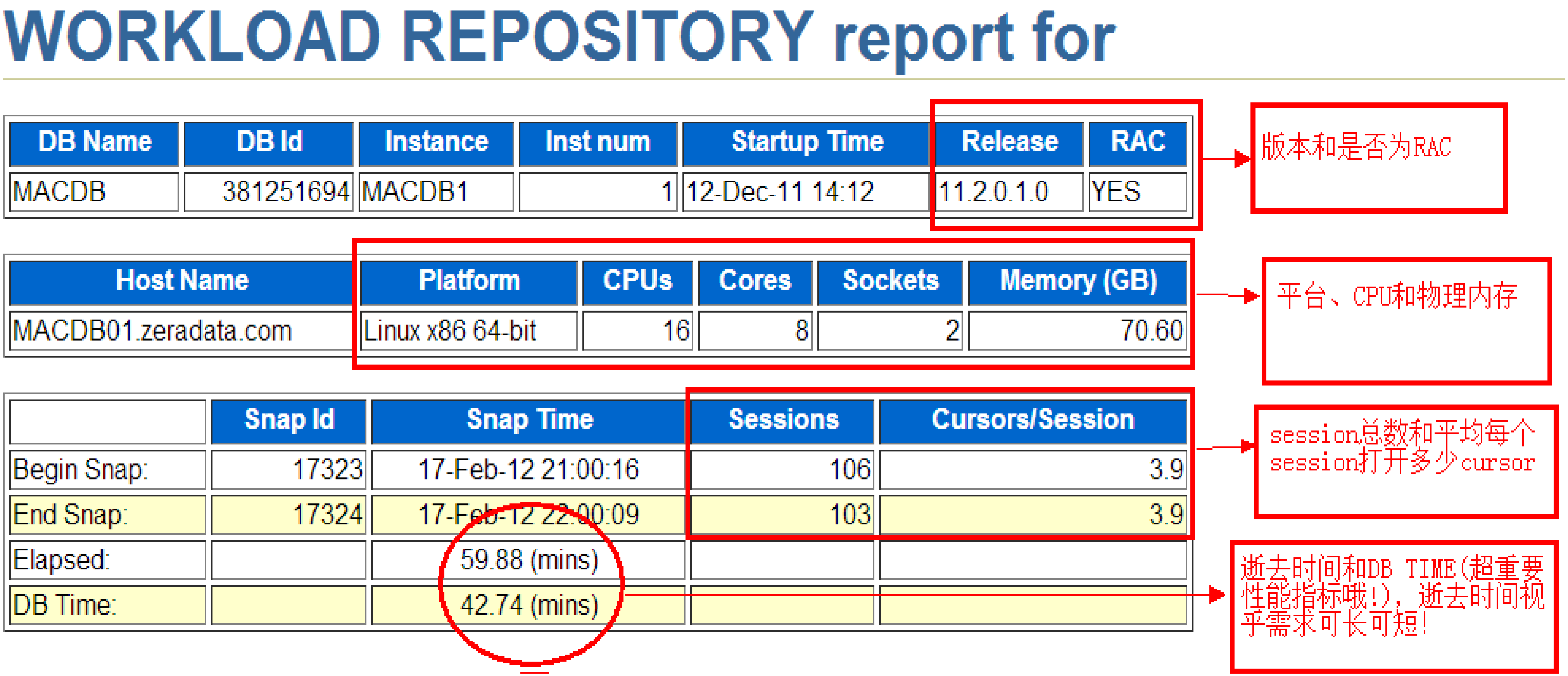


祇園精舎の鐘の聲
諸行無常の響きあり
娑羅雙樹の花の色
盛者必衰の理をあらはす
おごれる人も久しからず
唯春の夜の夢のごとし
たけき者も遂にはほろびぬ
偏に風の前の塵に同じ

性能无常

盛者不调优则必衰

DB TIME



WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
MACDB	381251694	MACDB1	1	12-Dec-11 14:12	11.2.0.1.0	YES

版本和是否为RAC

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
MACDB01.zeradata.com	Linux x86 64-bit	16	8	2	70.60

平台、CPU和物理内存

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	17323	17-Feb-12 21:00:16	106	3.9
End Snap:	17324	17-Feb-12 22:00:09	103	3.9
Elapsed:		59.88 (mins)		
DB Time:		42.74 (mins)		

session总数和平均每个session打开多少cursor

逝去时间和DB TIME(超重要性能指标哦!), 逝去时间视乎需求可长可短!

信息量很巨大哦！！

Elapsed快照逝去时间，如果为了诊断特定时段性能问题则不宜过长**15**分钟**~2**、**3**个小时。如果是看全天负载那么可以长一些。

最常见是**60**分钟后者**120**分钟，视乎实际需求，无成法。

Cursors/session ➔ open_cursors ORA-100诊断

DB TIME

DB TIME= 所有前台session花费在database调用上的总和时间

- 注意是前台进程**foreground sessions**
- 包括**CPU**时间、**IO Time**、和其他一系列非空闲等待时间，别忘了**cpu on queue time**

DB TIME 不等于 响应时间，**DB TIME**高了未必响应慢，**DB TIME**低了未必响应快

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 =60 min， 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 ➔ 系统**hang**了吧？

DB TIME= DB CPU + NON-Idle Wait + Wait on CPU queue

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

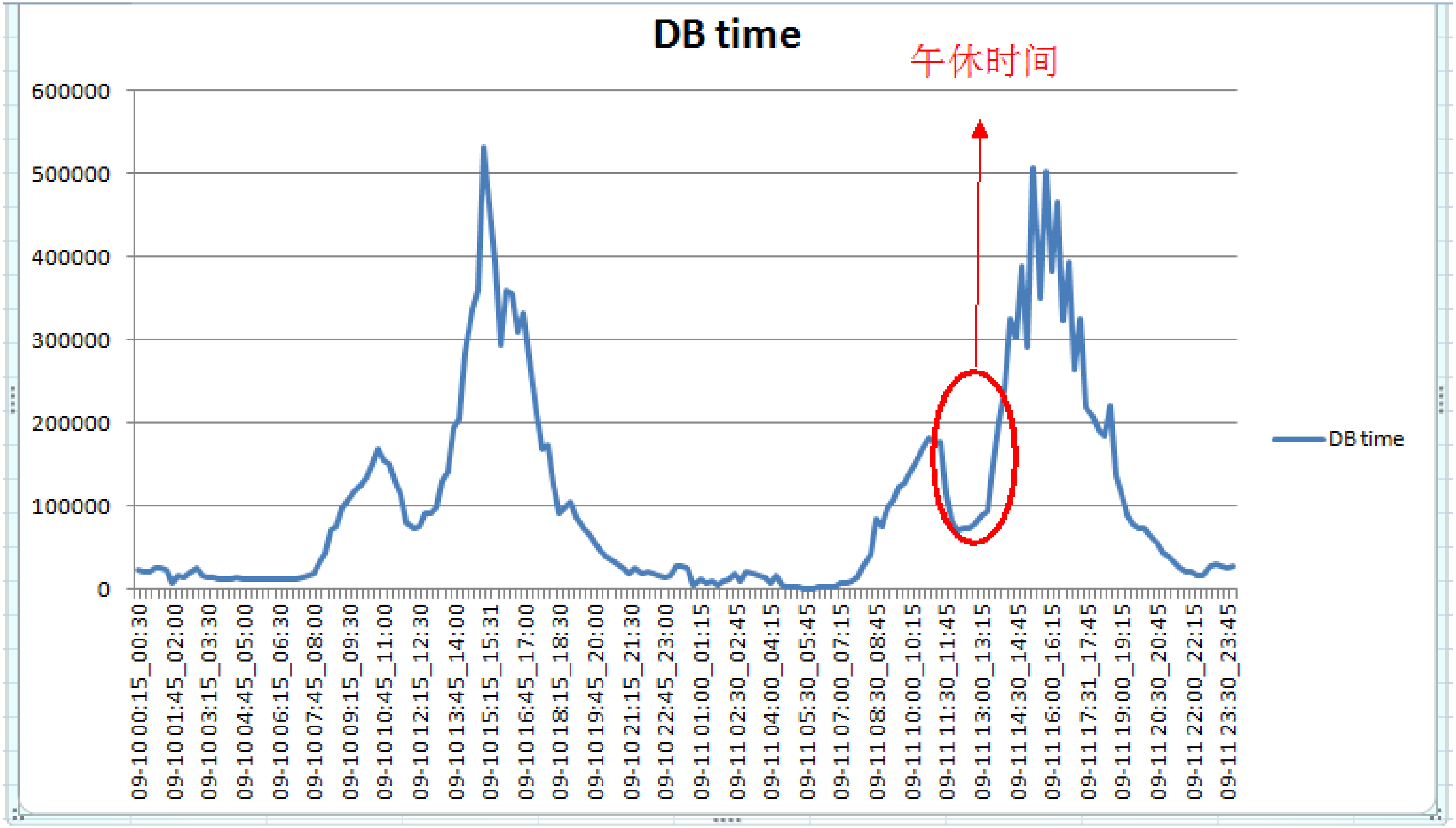
AAS= (120+ 60)/60=3 ➔ 主机**load** 亦为**3**，此时**vmstat** 看**waiting for run time**

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 + ……….** 阿猫阿狗

查最近7天的DB Time

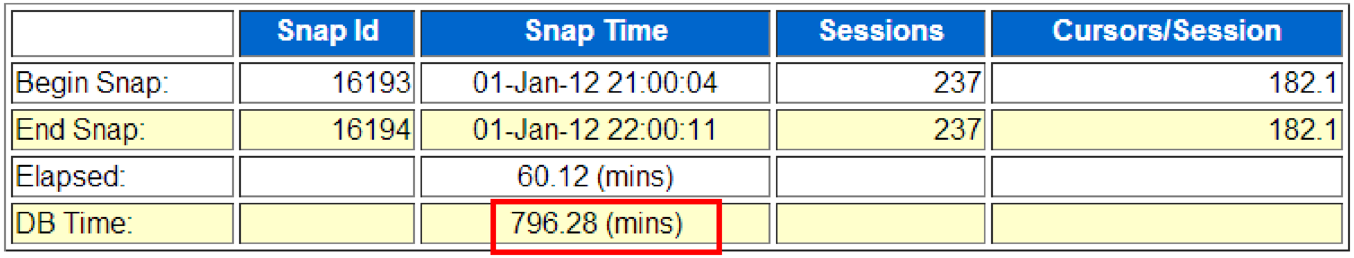
```
WITH sysstat AS
 (select sn.begin_interval_time begin_interval_time,
         sn.end_interval_time end_interval_time,
         ss.stat_name stat_name,
         ss.value e_value,
         lag(ss.value, 1) over(order by ss.snap_id) b_value
    from dba_hist_sysstat ss, dba_hist_snapshot sn
   where 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)) / (extract(day
  from(end_interval_time - begin_interval_time)) * 24 * 60 * 60 + extract(hour
  from(end_interval_time - begin_interval_time)) * 60 * 60 + extract(minute
  from(end_interval_time - begin_interval_time)) * 60 + extract(second
  from(end_interval_time - begin_interval_time))), 0) per_sec
  from sysstat
 where(e_value - nvl(b_value, 0)) > 0 and nvl(b_value, 0) > 0
 /
```

DB Time折线图：文似看山不喜平

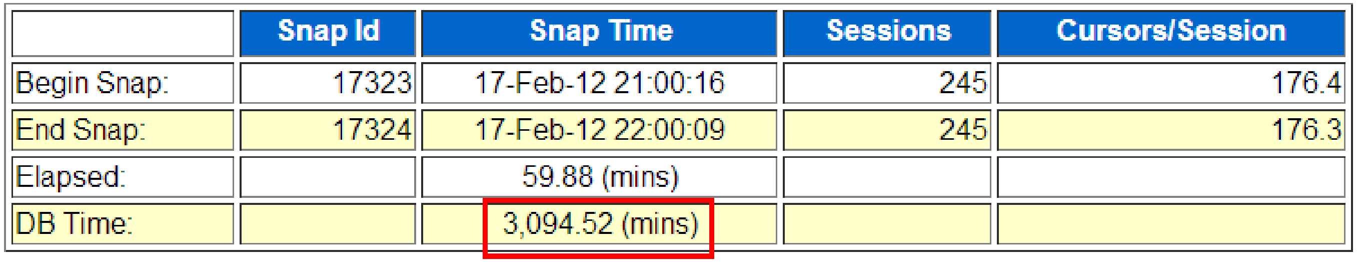


实战DB Time案例

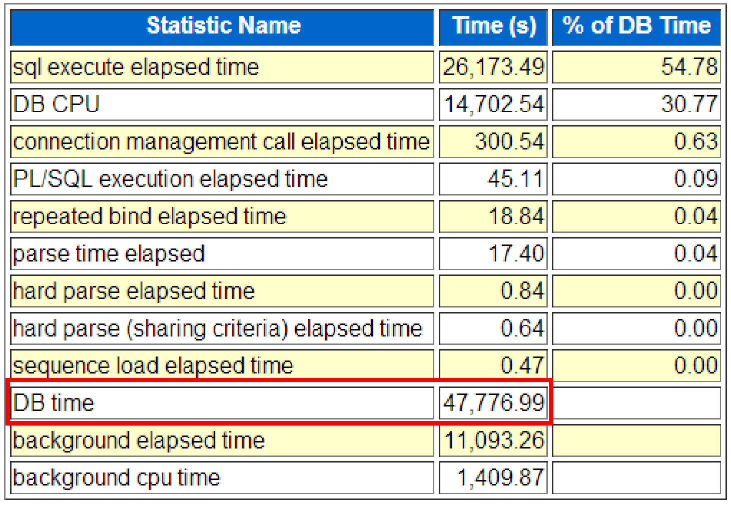
同一套库 1月 VS 2月



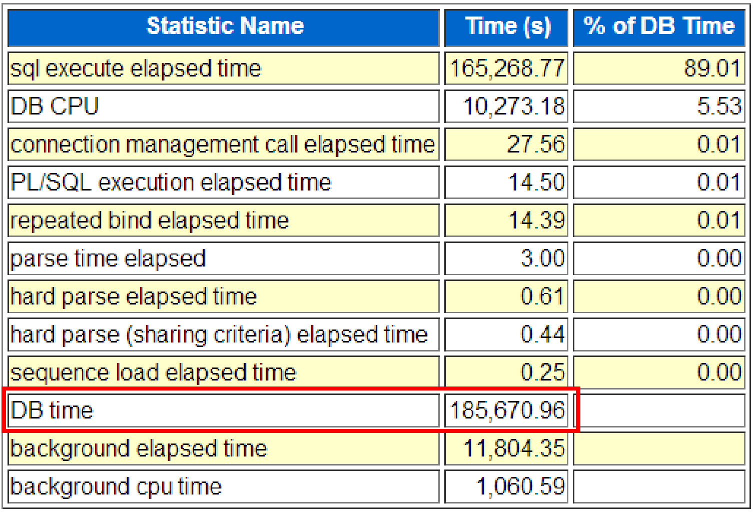
	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	16193	01-Jan-12 21:00:04	237	182.1
End Snap:	16194	01-Jan-12 22:00:11	237	182.1
Elapsed:		60.12 (mins)		
DB Time:		796.28 (mins)		



	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	17323	17-Feb-12 21:00:16	245	176.4
End Snap:	17324	17-Feb-12 22:00:09	245	176.3
Elapsed:		59.88 (mins)		
DB Time:		3,094.52 (mins)		



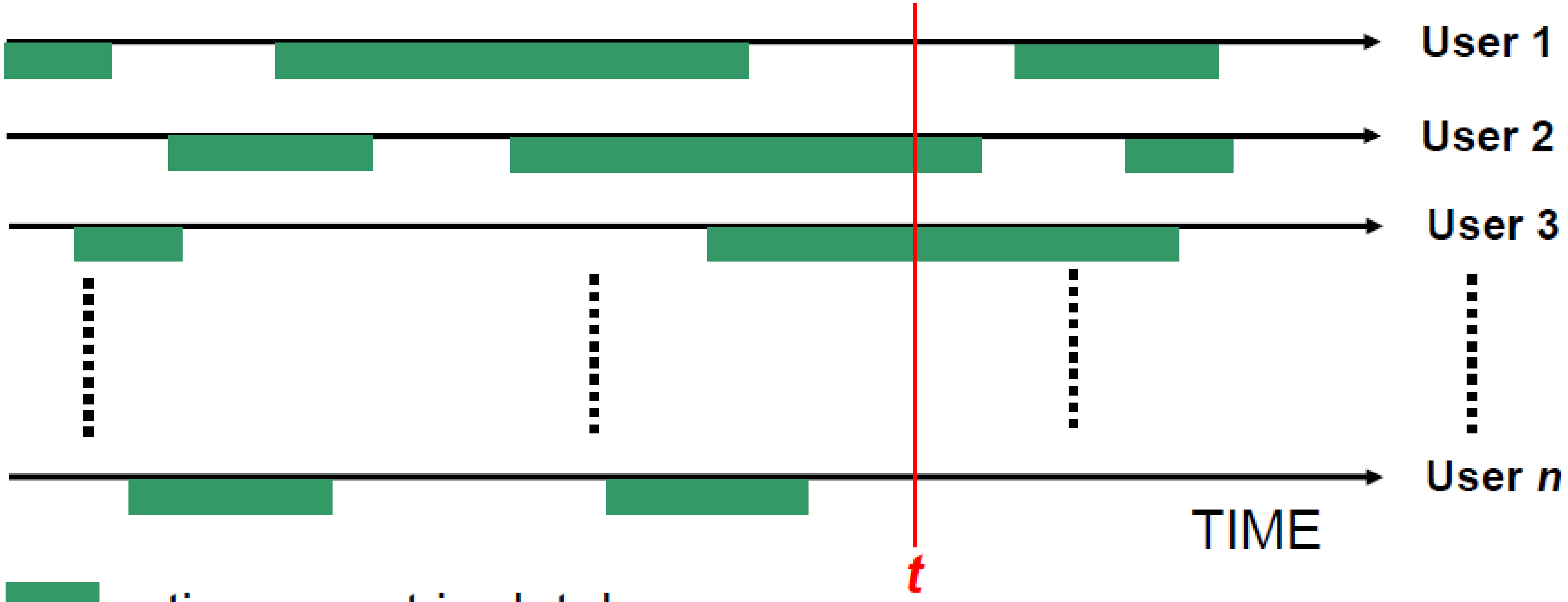
Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	26,173.49	54.78
DB CPU	14,702.54	30.77
connection management call elapsed time	300.54	0.63
PL/SQL execution elapsed time	45.11	0.09
repeated bind elapsed time	18.84	0.04
parse time elapsed	17.40	0.04
hard parse elapsed time	0.84	0.00
hard parse (sharing criteria) elapsed time	0.64	0.00
sequence load elapsed time	0.47	0.00
DB time	47,776.99	
background elapsed time	11,093.26	
background cpu time	1,409.87	



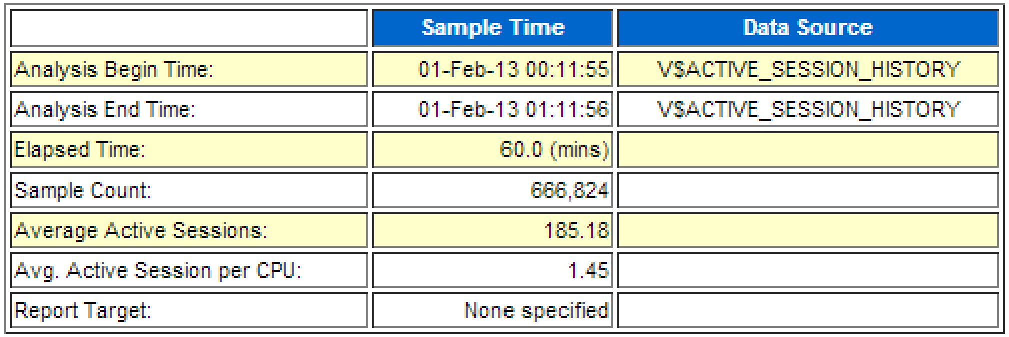
Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	165,268.77	89.01
DB CPU	10,273.18	5.53
connection management call elapsed time	27.56	0.01
PL/SQL execution elapsed time	14.50	0.01
repeated bind elapsed time	14.39	0.01
parse time elapsed	3.00	0.00
hard parse elapsed time	0.61	0.00
hard parse (sharing criteria) elapsed time	0.44	0.00
sequence load elapsed time	0.25	0.00
DB time	185,670.96	
background elapsed time	11,804.35	
background cpu time	1,060.59	

AAS平均活动会话==与ASH结合

At time *t* we have 2 active sessions



🟩 = time spent in database



	Sample Time	Data Source
Analysis Begin Time:	01-Feb-13 00:11:55	V$ACTIVE_SESSION_HISTORY
Analysis End Time:	01-Feb-13 01:11:56	V$ACTIVE_SESSION_HISTORY
Elapsed Time:	60.0 (mins)	
Sample Count:	666,824	
Average Active Sessions:	185.18	
Avg. Active Session per CPU:	1.45	
Report Target:	None specified	

ASH报告开篇即为AAS

Cache Size 兵马未动，缓存先行

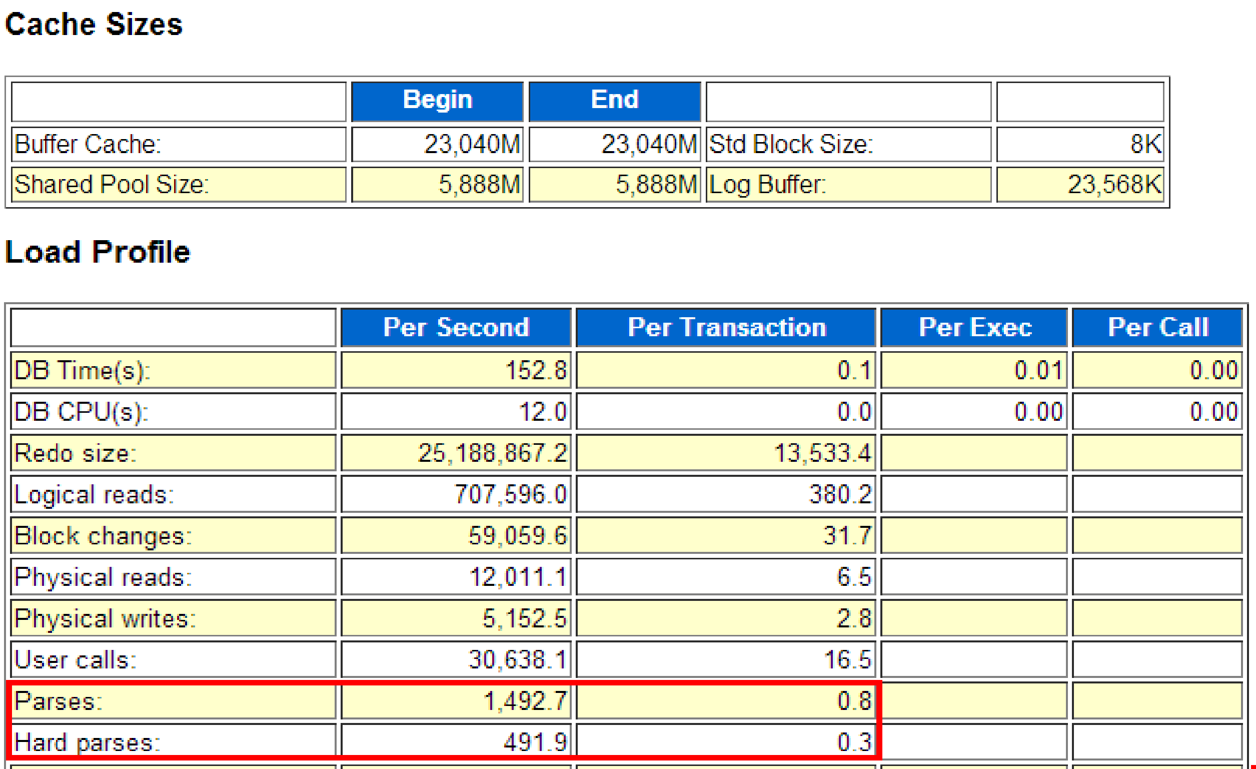
内存管理方式：**MSMM、ASMM(sga_target)、AMM(memory_target)**

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！

这里说 shared pool一直收缩，则在shrink过程中一些row cache 对象被lock住可能导致前台row cache lock等解析等待，最后别让shared pool shrink。

如果这里shared pool一直在grow，那说明shared pool原有大小不足以满足需求(可能是大量硬解析)，结合下文的解析信息和SGA breakdown来一起诊断问题。

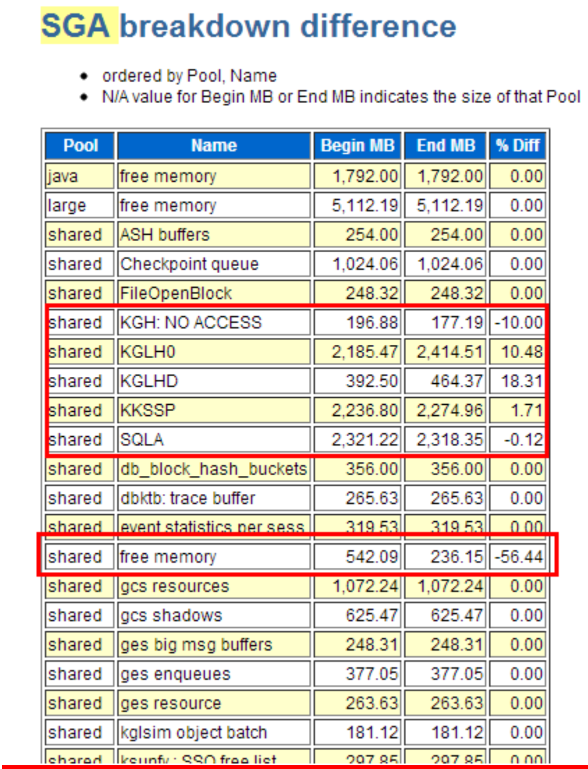


Cache Sizes

	Begin	End		
Buffer Cache:	23,040M	23,040M	Std Block Size:	8K
Shared Pool Size:	5,888M	5,888M	Log Buffer:	23,568K

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	152.8	0.1	0.01	0.00
DB CPU(s):	12.0	0.0	0.00	0.00
Redo size:	25,188,867.2	13,533.4		
Logical reads:	707,596.0	380.2		
Block changes:	59,059.6	31.7		
Physical reads:	12,011.1	6.5		
Physical writes:	5,152.5	2.8		
User calls:	30,638.1	16.5		
Parses:	1,492.7	0.8		
Hard parses:	491.9	0.3		

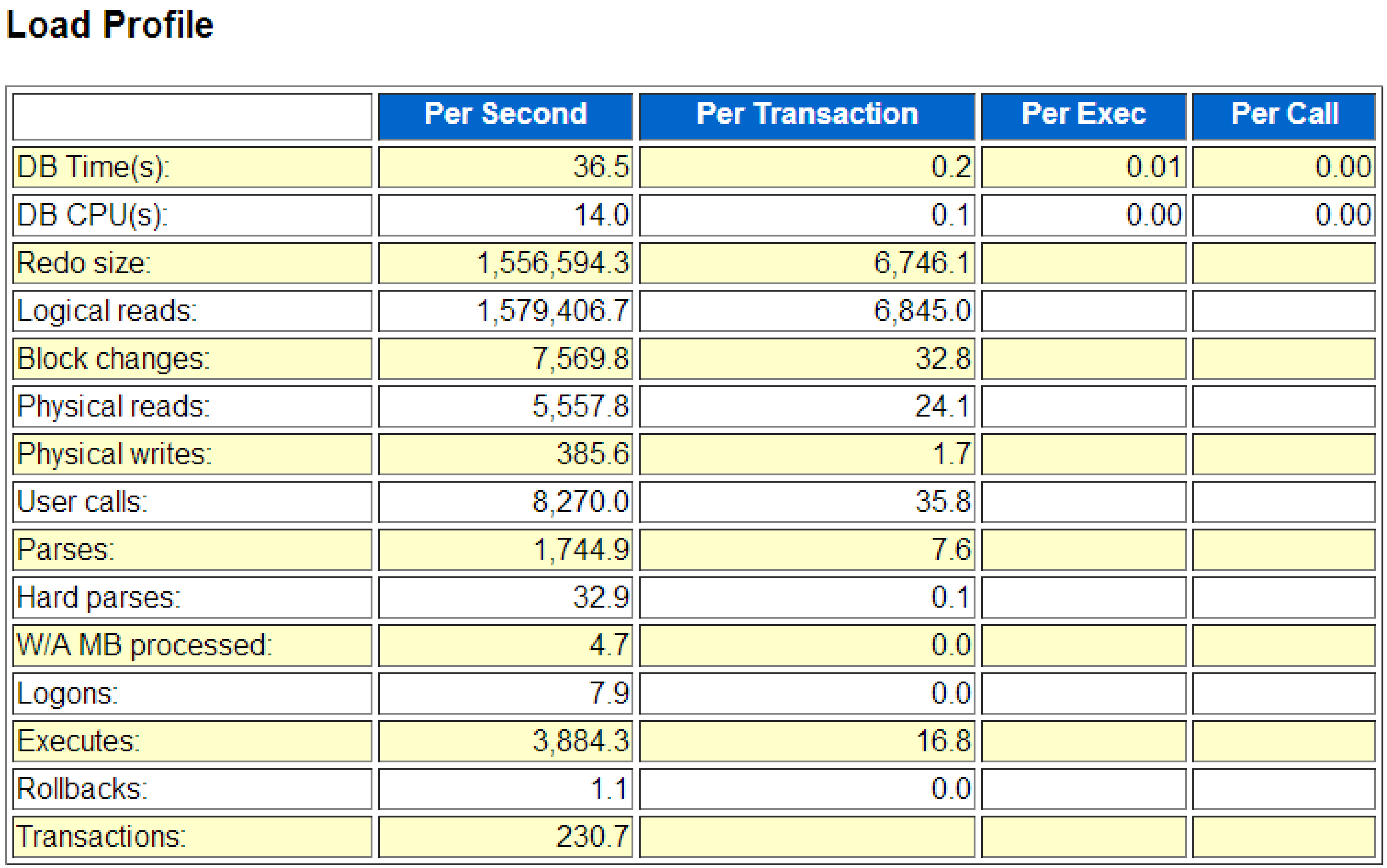


SGA breakdown difference

- ordered by Pool, Name
- N/A value for Begin MB or End MB indicates the size of that Pool

Pool	Name	Begin MB	End MB	% Diff
java	free memory	1,792.00	1,792.00	0.00
large	free memory	5,112.19	5,112.19	0.00
shared	ASH buffers	254.00	254.00	0.00
shared	Checkpoint queue	1,024.06	1,024.06	0.00
shared	FileOpenBlock	248.32	248.32	0.00
shared	KGH: NO ACCESS	196.88	177.19	-10.00
shared	KGLH0	2,185.47	2,414.51	10.48
shared	KGLHD	392.50	464.37	18.31
shared	KKSSP	2,236.80	2,274.96	1.71
shared	SQLA	2,321.22	2,318.35	-0.12
shared	db_block_hash_buckets	356.00	356.00	0.00
shared	dbktb: trace buffer	265.63	265.63	0.00
shared	event statistics per sess	319.53	319.53	0.00
shared	free memory	542.09	236.15	-56.44
shared	gcs resources	1,072.24	1,072.24	0.00
shared	gcs shadows	625.47	625.47	0.00
shared	ges big msg buffers	248.31	248.31	0.00
shared	ges enqueues	377.05	377.05	0.00
shared	ges resource	263.63	263.63	0.00
shared	kglsim object batch	181.12	181.12	0.00
shared	ksunfv : SSO free list	297.85	297.85	0.00

Load Profile



Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	36.5	0.2	0.01	0.00
DB CPU(s):	14.0	0.1	0.00	0.00
Redo size:	1,556,594.3	6,746.1		
Logical reads:	1,579,406.7	6,845.0		
Block changes:	7,569.8	32.8		
Physical reads:	5,557.8	24.1		
Physical writes:	385.6	1.7		
User calls:	8,270.0	35.8		
Parses:	1,744.9	7.6		
Hard parses:	32.9	0.1		
W/A MB processed:	4.7	0.0		
Logons:	7.9	0.0		
Executes:	3,884.3	16.8		
Rollbacks:	1.1	0.0		
Transactions:	230.7			

信息量太大！！

Load Profile

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。

Load Profile

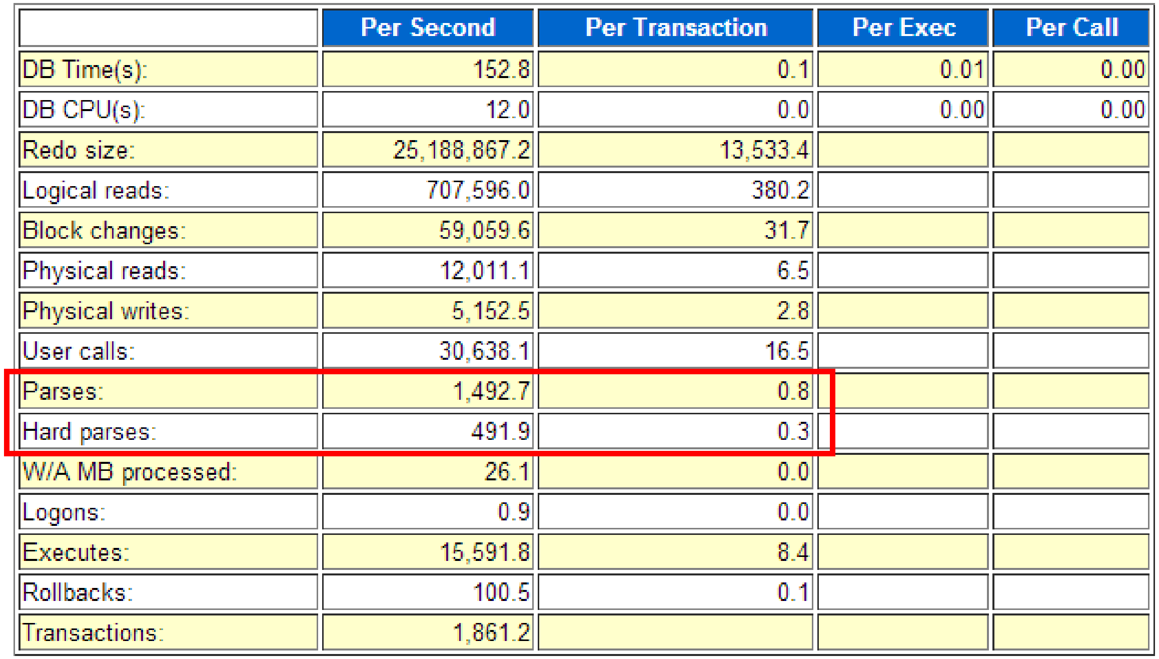
Physical writes单位 次数*块数，主要是DBWR写datafile，也有direct path write。 dbwr长期写出慢会导致定期log file switch(checkpoint no complete) 检查点无法完成的前台等待。

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

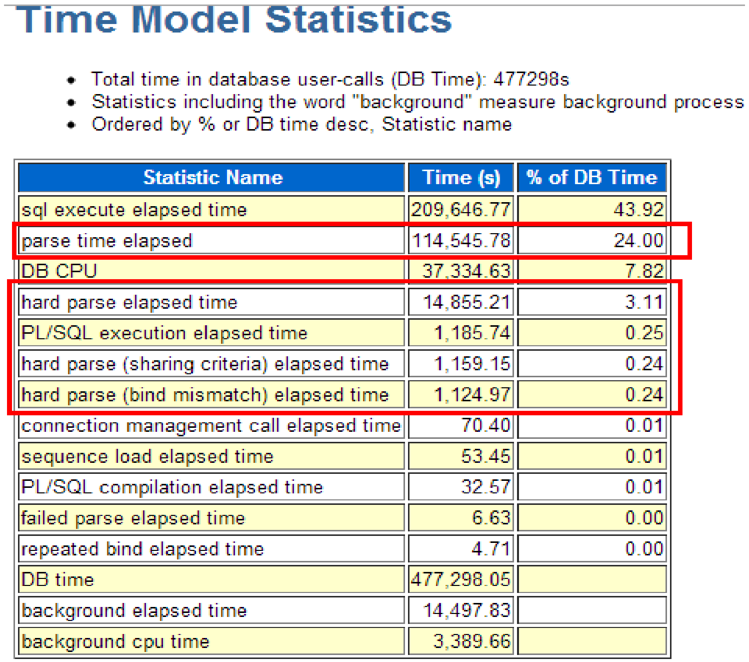
Parses，解析次数，包括软解析+硬解析，软解析优化得不好，则夸张地说几乎等于每秒SQL执行次数。 即执行解析比1:1，而我们希望的是 解析一次 到处运行哦！

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

Hard Parses实战案例



	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	152.8	0.1	0.01	0.00
DB CPU(s):	12.0	0.0	0.00	0.00
Redo size:	25,188,867.2	13,533.4		
Logical reads:	707,596.0	380.2		
Block changes:	59,059.6	31.7		
Physical reads:	12,011.1	6.5		
Physical writes:	5,152.5	2.8		
User calls:	30,638.1	16.5		
Parses:	1,492.7	0.8		
Hard parses:	491.9	0.3		
W/A MB processed:	26.1	0.0		
Logons:	0.9	0.0		
Executes:	15,591.8	8.4		
Rollbacks:	100.5	0.1		
Transactions:	1,861.2			



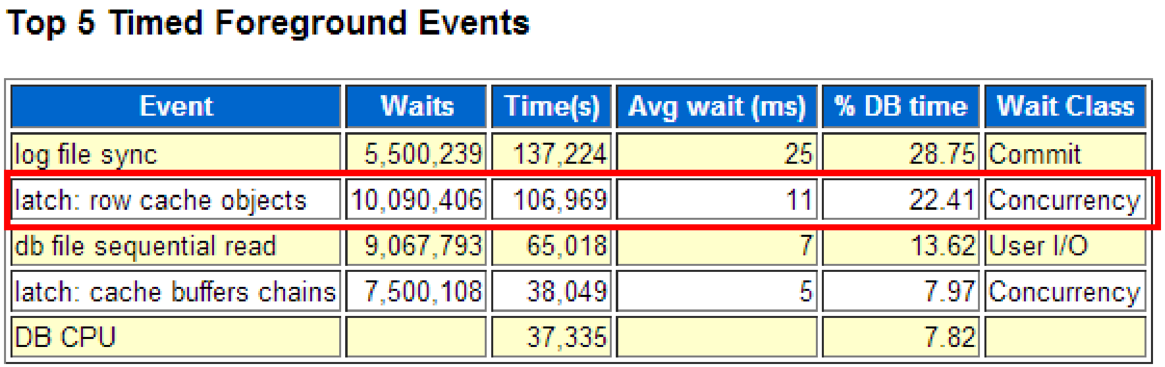
Time Model Statistics

- Total time in database user-calls (DB Time): 477298s
- Statistics including the word "background" measure background process
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	209,646.77	43.92
parse time elapsed	114,545.78	24.00
DB CPU	37,334.63	7.82
hard parse elapsed time	14,855.21	3.11
PL/SQL execution elapsed time	1,185.74	0.25
hard parse (sharing criteria) elapsed time	1,159.15	0.24
hard parse (bind mismatch) elapsed time	1,124.97	0.24
connection management call elapsed time	70.40	0.01
sequence load elapsed time	53.45	0.01
PL/SQL compilation elapsed time	32.57	0.01
failed parse elapsed time	6.63	0.00
repeated bind elapsed time	4.71	0.00
DB time	477,298.05	
background elapsed time	14,497.83	
background cpu time	3,389.66	

硬解析数和 hard parse elapsed time对应， 看一眼Time Model Statistics，即可知该系统是否 是 解析敏感的

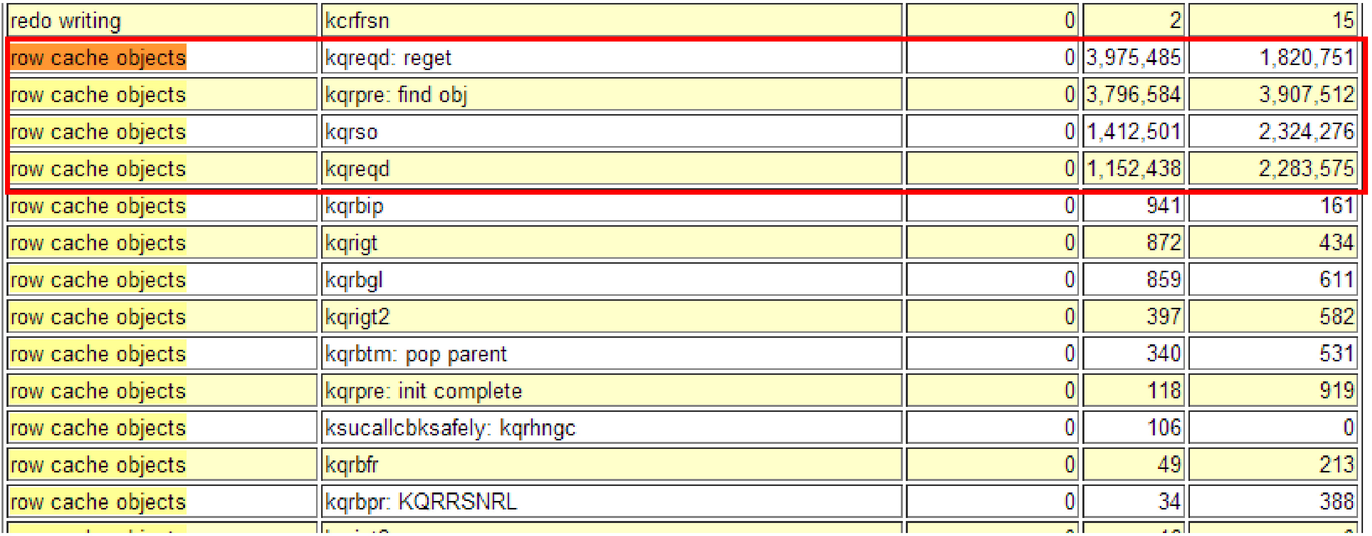
Hard Parses实战案例



Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
log file sync	5,500,239	137,224	25	28.75	Commit
latch: row cache objects	10,090,406	106,969	11	22.41	Concurrency
db file sequential read	9,067,793	65,018	7	13.62	User I/O
latch: cache buffers chains	7,500,108	38,049	5	7.97	Concurrency
DB CPU		37,335		7.82	

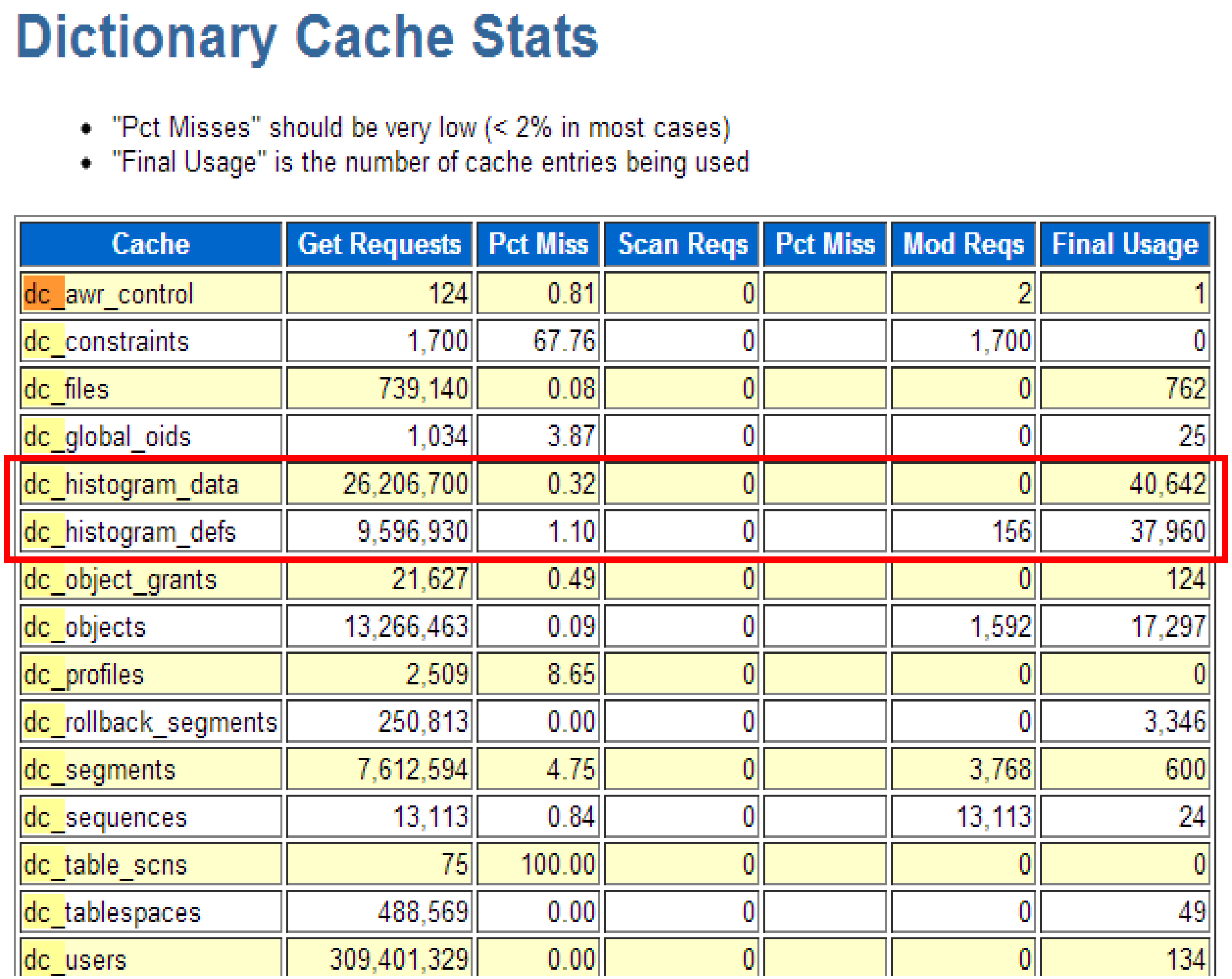
Latch:row cache objects保护row cache state object，例如dc_segments、dc_users



redo writing	kcrfrsn	0	2	15
row cache objects	kqreqd: reget	0	3,975,485	1,820,751
row cache objects	kqrpre: find obj	0	3,796,584	3,907,512
row cache objects	kqrso	0	1,412,501	2,324,276
row cache objects	kqreqd	0	1,152,438	2,283,575
row cache objects	kqrbip	0	941	161
row cache objects	kqrigt	0	872	434
row cache objects	kqrbgl	0	859	611
row cache objects	kqrigt2	0	397	582
row cache objects	kqrbtm: pop parent	0	340	531
row cache objects	kqrpre: init complete	0	118	919
row cache objects	ksucallcbksafely: kqrhngc	0	106	0
row cache objects	kqrbfr	0	49	213
row cache objects	kqrbpr: KQRRSNRL	0	34	388

kqrpo parent cache object header
kqrso subordinate cache header
kqrpre() interface to get a row cache object

Hard Parses实战案例



Dictionary Cache Stats

- "Pct Misses" should be very low (< 2% in most cases)
- "Final Usage" is the number of cache entries being used

Cache	Get Requests	Pct Miss	Scan Reqs	Pct Miss	Mod Reqs	Final Usage
dc_awr_control	124	0.81	0		2	1
dc_constraints	1,700	67.76	0		1,700	0
dc_files	739,140	0.08	0		0	762
dc_global_oids	1,034	3.87	0		0	25
dc_histogram_data	26,206,700	0.32	0		0	40,642
dc_histogram_defs	9,596,930	1.10	0		156	37,960
dc_object_grants	21,627	0.49	0		0	124
dc_objects	13,266,463	0.09	0		1,592	17,297
dc_profiles	2,509	8.65	0		0	0
dc_rollback_segments	250,813	0.00	0		0	3,346
dc_segments	7,612,594	4.75	0		3,768	600
dc_sequences	13,113	0.84	0		13,113	24
dc_table_scns	75	100.00	0		0	0
dc_tablespaces	488,569	0.00	0		0	49
dc_users	309,401,329	0.00	0		0	134

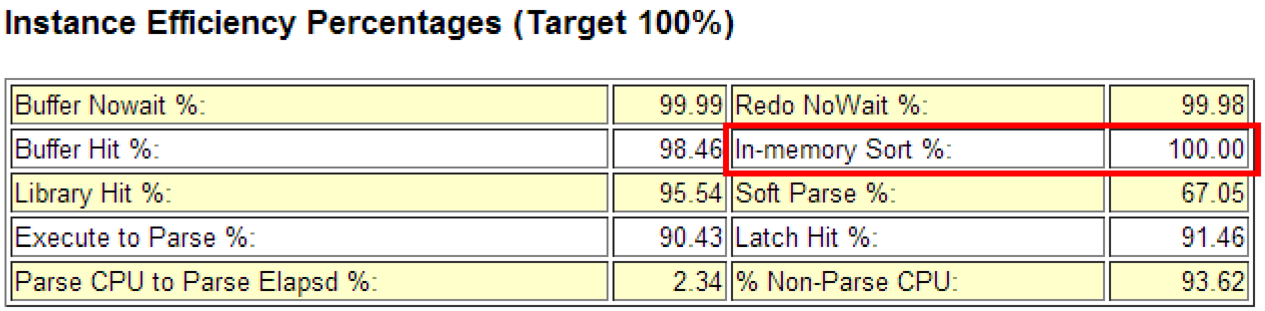
Dc_histogram，字典缓存*row cache*中的 直方图信息

大量硬解析=》直方图字典缓存争用

思路：解决硬解析，或者减少*dc_histogram*争用

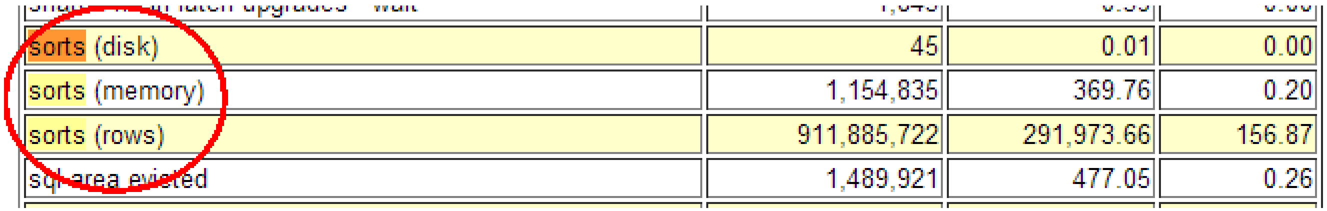
W/A MB processed

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



Instance Efficiency Percentages (Target 100%)

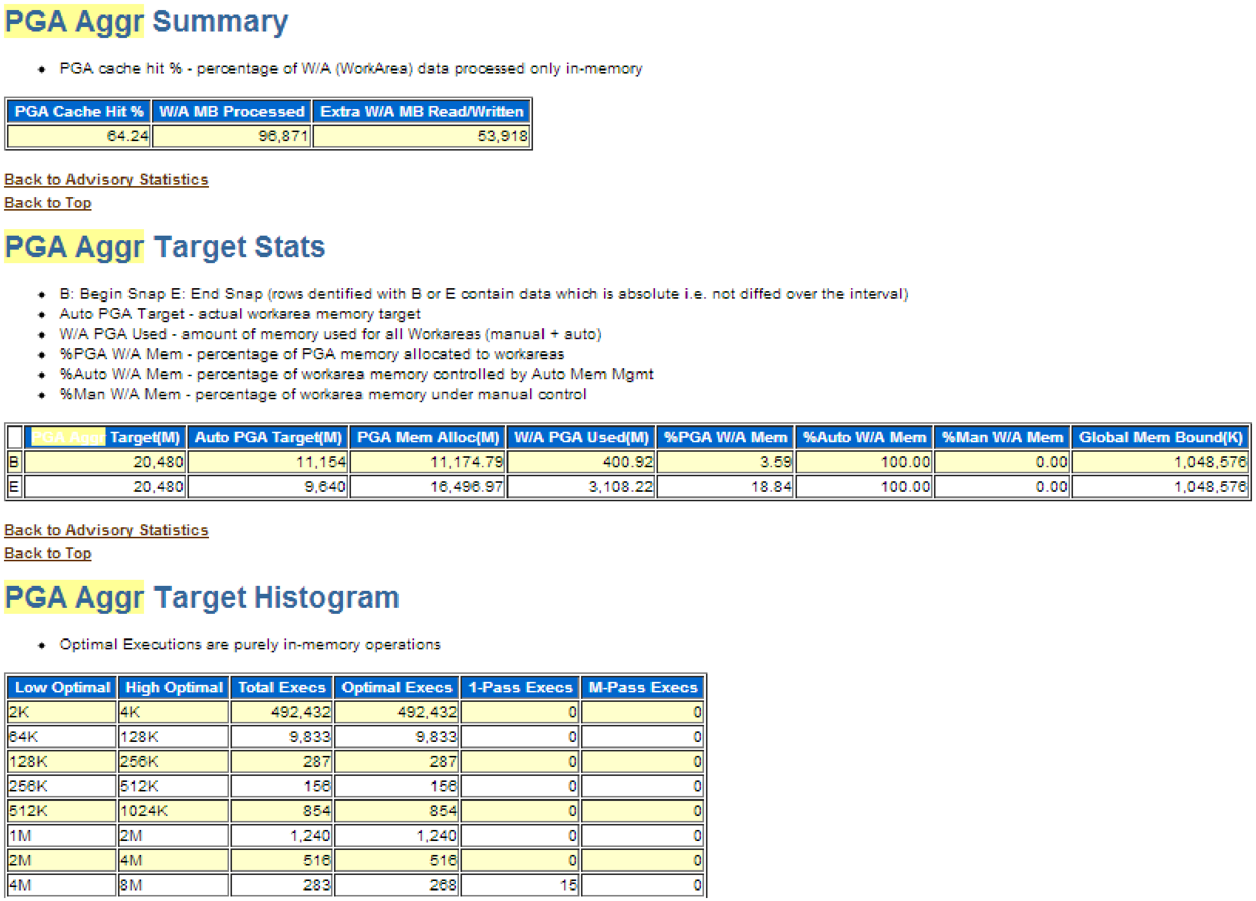
Buffer Nowait %:	99.99	Redo NoWait %:	99.98
Buffer Hit %:	98.46	In-memory Sort %:	100.00
Library Hit %:	95.54	Soft Parse %:	67.05
Execute to Parse %:	90.43	Latch Hit %:	91.46
Parse CPU to Parse Elapsd %:	2.34	% Non-Parse CPU:	93.62



sorts (disk)	45	0.01	0.00
sorts (memory)	1,154,835	369.76	0.20
sorts (rows)	911,885,722	291,973.66	156.87
sql area evicted	1,489,921	477.05	0.26

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 Aggr Summary

- PGA cache hit % - percentage of W/A (WorkArea) data processed only in-memory

PGA Cache Hit %	W/A MB Processed	Extra W/A MB Read/Written
64.24	96,871	53,918

Back to Advisory Statistics
Back to Top

PGA Aggr Target Stats

- B: Begin Snap E: End Snap (rows dentified with B or E contain data which is absolute i.e. not diffed over the interval)
- Auto PGA Target - actual workarea memory target
- W/A PGA Used - amount of memory used for all Workareas (manual + auto)
- %PGA W/A Mem - percentage of PGA memory allocated to workareas
- %Auto W/A Mem - percentage of workarea memory controlled by Auto Mem Mgmt
- %Man W/A Mem - percentage of workarea memory under manual control

	PGA Aggr Target(M)	Auto PGA Target(M)	PGA Mem Alloc(M)	W/A PGA Used(M)	%PGA W/A Mem	%Auto W/A Mem	%Man W/A Mem	Global Mem Bound(K)
B	20,480	11,154	11,174.79	400.92	3.59	100.00	0.00	1,048,576
E	20,480	9,640	16,496.97	3,108.22	18.84	100.00	0.00	1,048,576

Back to Advisory Statistics
Back to Top

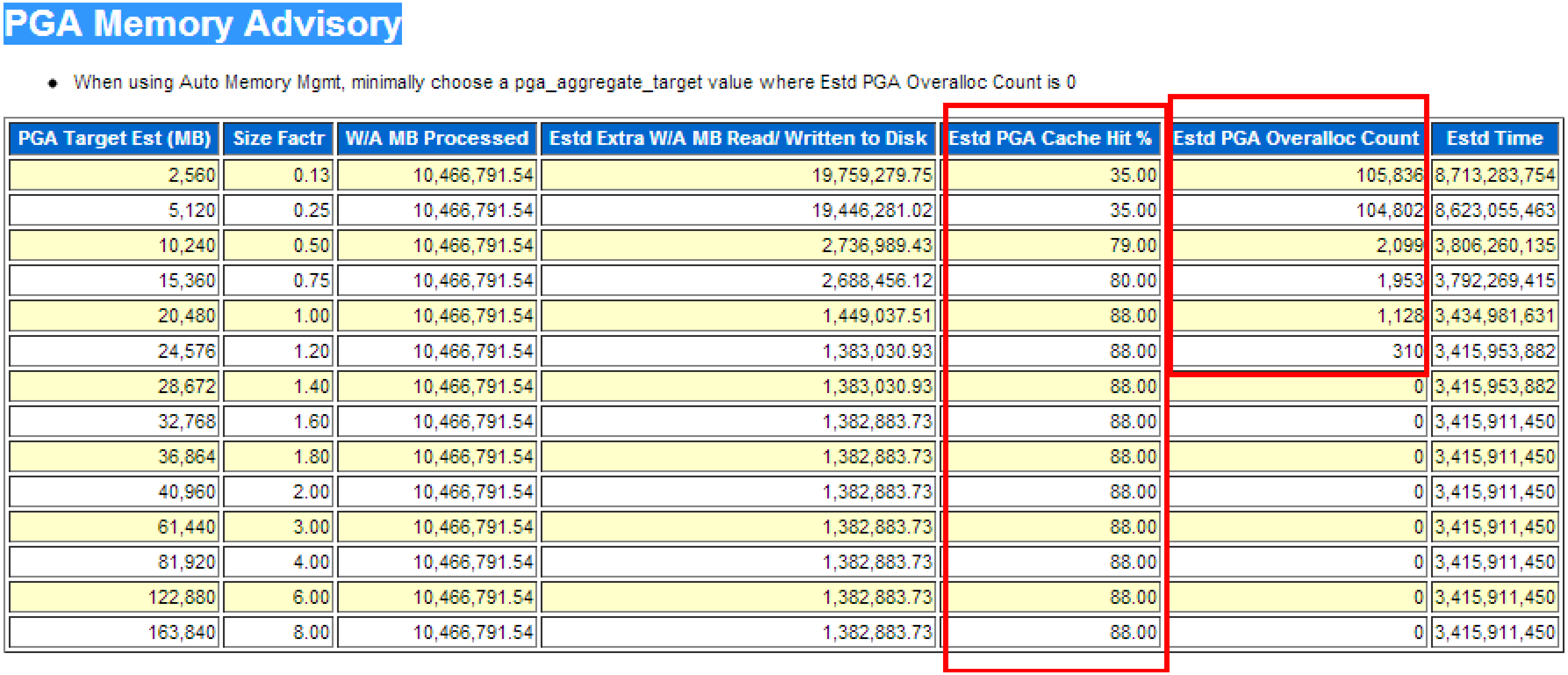
PGA Aggr Target Histogram

- Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
2K	4K	492,432	492,432	0	0
64K	128K	9,833	9,833	0	0
128K	256K	287	287	0	0
256K	512K	156	156	0	0
512K	1024K	854	854	0	0
1M	2M	1,240	1,240	0	0
2M	4M	516	516	0	0
4M	8M	283	268	15	0

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%



PGA Memory Advisory

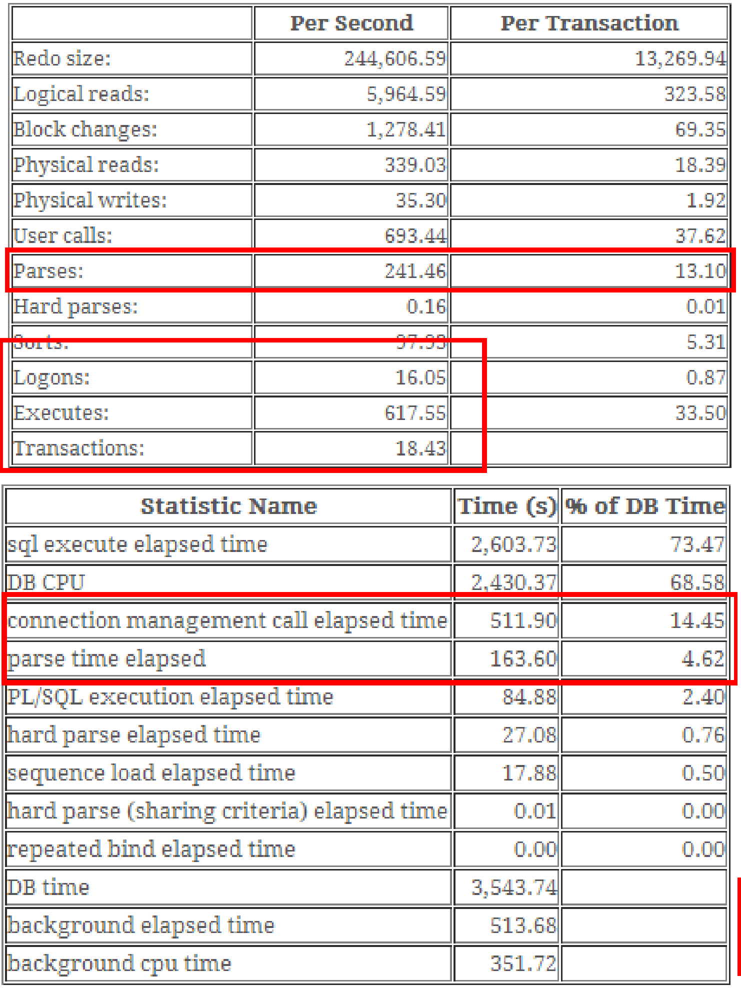
- When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

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	Estd Time
2,560	0.13	10,466,791.54	19,759,279.75	35.00	105,836	8,713,283,754
5,120	0.25	10,466,791.54	19,446,281.02	35.00	104,802	8,623,055,463
10,240	0.50	10,466,791.54	2,736,989.43	79.00	2,099	3,806,260,135
15,360	0.75	10,466,791.54	2,688,456.12	80.00	1,953	3,792,269,415
20,480	1.00	10,466,791.54	1,449,037.51	88.00	1,128	3,434,981,631
24,576	1.20	10,466,791.54	1,383,030.93	88.00	310	3,415,953,882
28,672	1.40	10,466,791.54	1,383,030.93	88.00	0	3,415,953,882
32,768	1.60	10,466,791.54	1,382,883.73	88.00	0	3,415,911,450
36,864	1.80	10,466,791.54	1,382,883.73	88.00	0	3,415,911,450
40,960	2.00	10,466,791.54	1,382,883.73	88.00	0	3,415,911,450
61,440	3.00	10,466,791.54	1,382,883.73	88.00	0	3,415,911,450
81,920	4.00	10,466,791.54	1,382,883.73	88.00	0	3,415,911,450
122,880	6.00	10,466,791.54	1,382,883.73	88.00	0	3,415,911,450
163,840	8.00	10,466,791.54	1,382,883.73	88.00	0	3,415,911,450

Load Profile- Logons 登陆

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

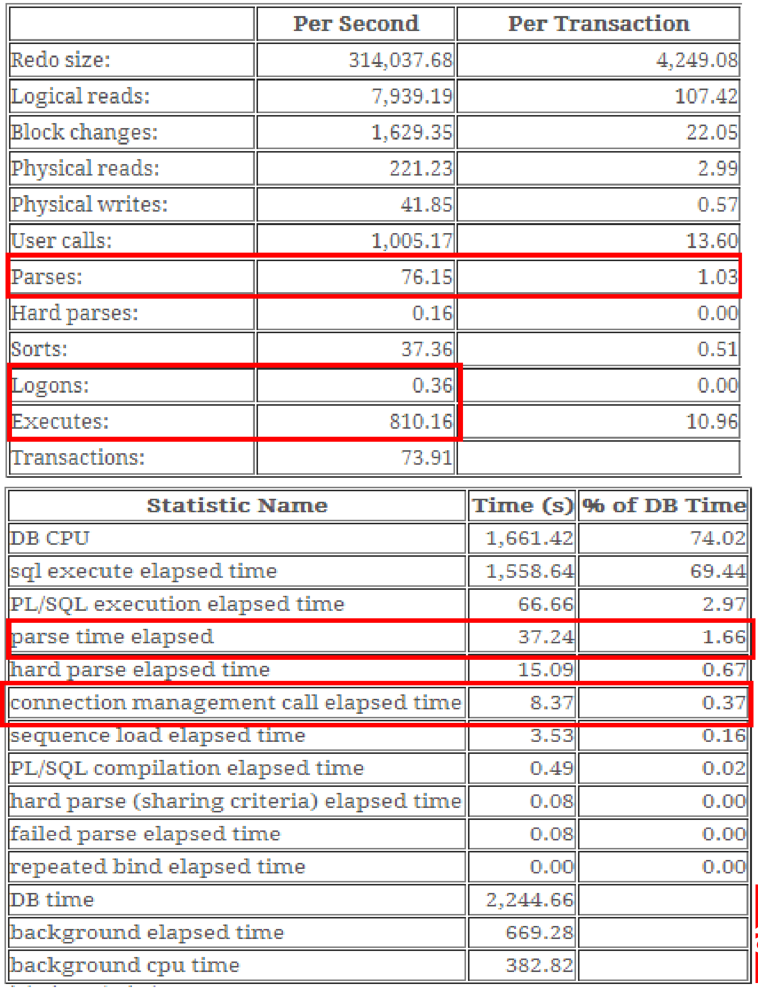
变更前



	Per Second	Per Transaction
Redo size:	244,606.59	13,269.94
Logical reads:	5,964.59	323.58
Block changes:	1,278.41	69.35
Physical reads:	339.03	18.39
Physical writes:	35.30	1.92
User calls:	693.44	37.62
Parses:	241.46	13.10
Hard parses:	0.16	0.01
Sorts:	97.93	5.31
Logons:	16.05	0.87
Executes:	617.55	33.50
Transactions:	18.43	

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	2,603.73	73.47
DB CPU	2,430.37	68.58
connection management call elapsed time	511.90	14.45
parse time elapsed	163.60	4.62
PL/SQL execution elapsed time	84.88	2.40
hard parse elapsed time	27.08	0.76
sequence load elapsed time	17.88	0.50
hard parse (sharing criteria) elapsed time	0.01	0.00
repeated bind elapsed time	0.00	0.00
DB time	3,543.74	
background elapsed time	513.68	
background cpu time	351.72	

变更后



	Per Second	Per Transaction
Redo size:	314,037.68	4,249.08
Logical reads:	7,939.19	107.42
Block changes:	1,629.35	22.05
Physical reads:	221.23	2.99
Physical writes:	41.85	0.57
User calls:	1,005.17	13.60
Parses:	76.15	1.03
Hard parses:	0.16	0.00
Sorts:	37.36	0.51
Logons:	0.36	0.00
Executes:	810.16	10.96
Transactions:	73.91	

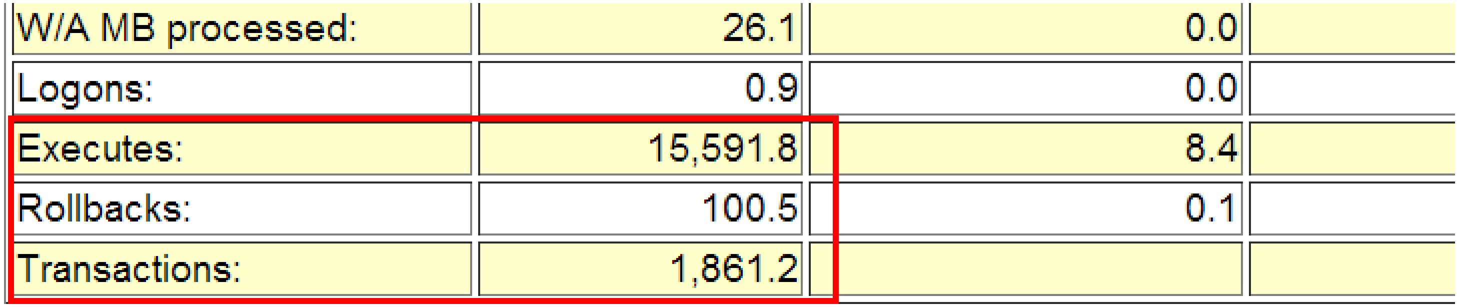
Statistic Name	Time (s)	% of DB Time
DB CPU	1,661.42	74.02
sql execute elapsed time	1,558.64	69.44
PL/SQL execution elapsed time	66.66	2.97
parse time elapsed	37.24	1.66
hard parse elapsed time	15.09	0.67
connection management call elapsed time	8.37	0.37
sequence load elapsed time	3.53	0.16
PL/SQL compilation elapsed time	0.49	0.02
hard parse (sharing criteria) elapsed time	0.08	0.00
failed parse elapsed time	0.08	0.00
repeated bind elapsed time	0.00	0.00
DB time	2,244.66	
background elapsed time	669.28	
background cpu time	382.82	

Load Profile-Execute、rollback、transactions

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

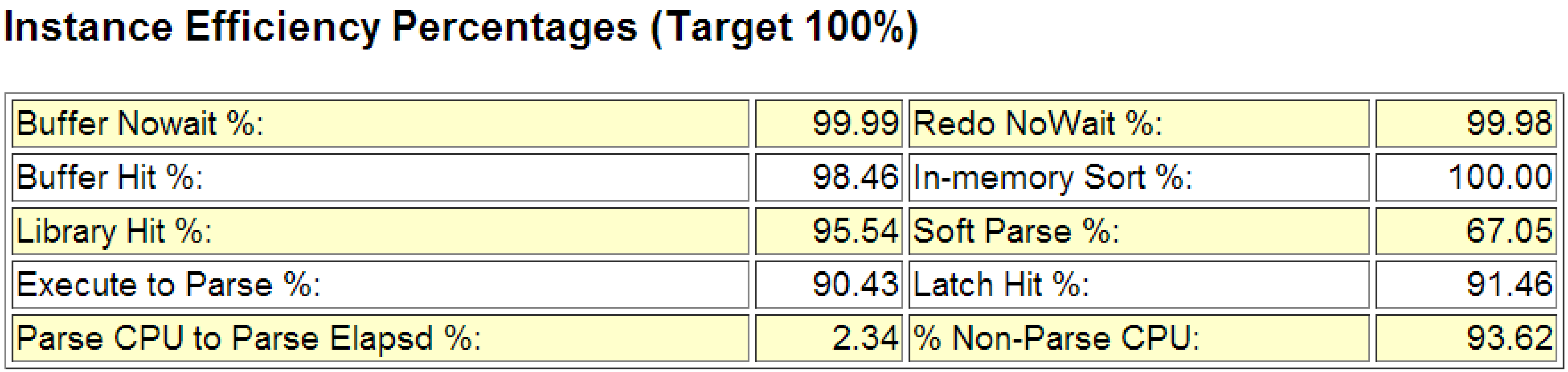
Transactions 每秒事务数，和数据库层的**TPS**，可以看做压力测试或比对性能是的一个指标，孤立看无意义。



W/A MB processed:	26.1	0.0	
Logons:	0.9	0.0	
Executes:	15,591.8	8.4	
Rollbacks:	100.5	0.1	
Transactions:	1,861.2		

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



Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.99	Redo NoWait %:	99.98
Buffer Hit %:	98.46	In-memory Sort %:	100.00
Library Hit %:	95.54	Soft Parse %:	67.05
Execute to Parse %:	90.43	Latch Hit %:	91.46
Parse CPU to Parse Elapsd %:	2.34	% Non-Parse CPU:	93.62

全部是越高越好！

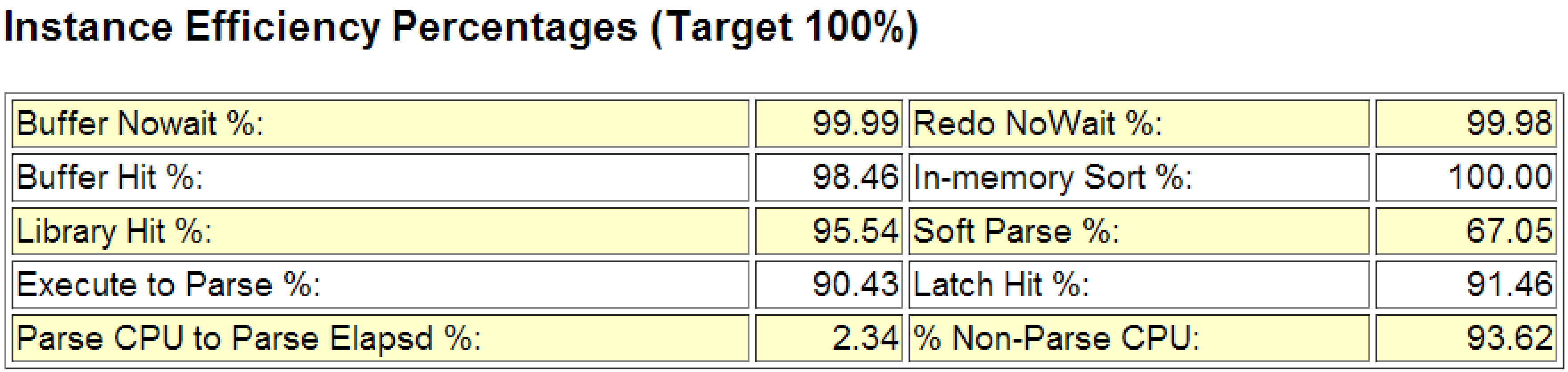
Buffer nowait%: session申请一个buffer(兼容模式)不等待的次数比例。

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

Instance Efficiency Percentages (Target 100%) In-Memory Sort%

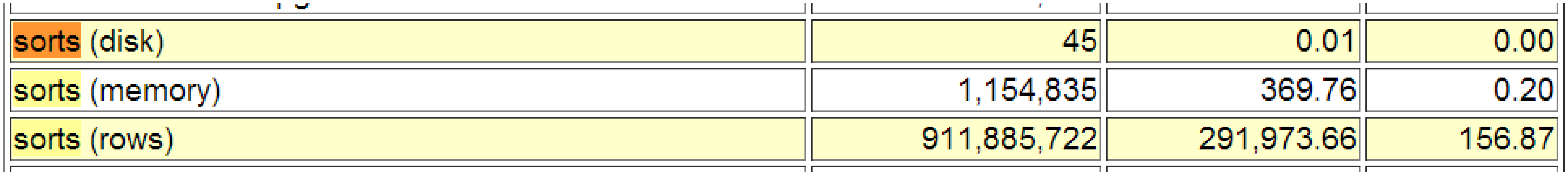
基于命中率的调优方法论已经过时，但仍具有参考价值



Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.99	Redo NoWait %:	99.98
Buffer Hit %:	98.46	In-memory Sort %:	100.00
Library Hit %:	95.54	Soft Parse %:	67.05
Execute to Parse %:	90.43	Latch Hit %:	91.46
Parse CPU to Parse Elapsd %:	2.34	% Non-Parse CPU:	93.62

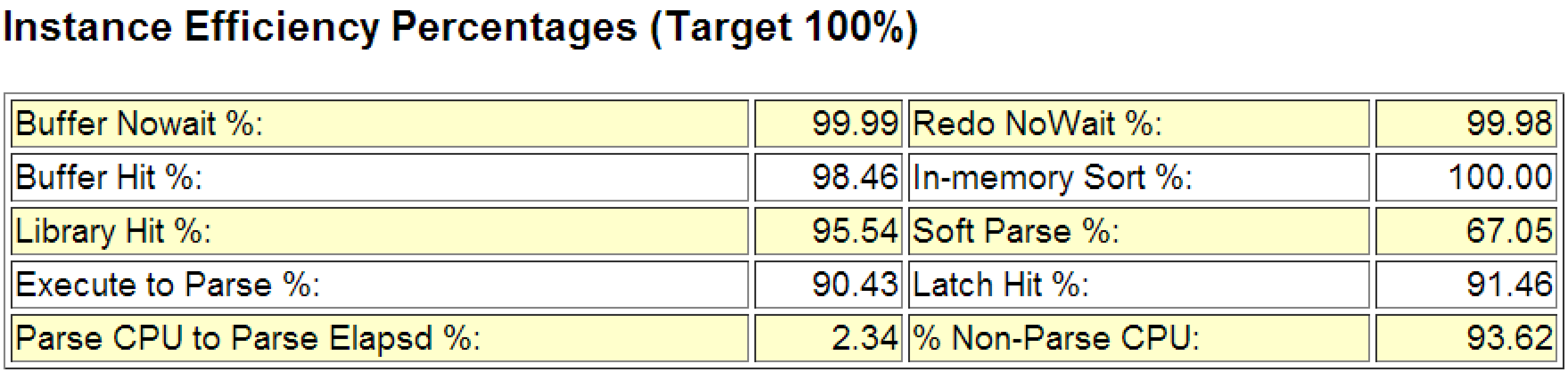
In-memory Sort%:这个指标因为它不计算workarea中所有的操作类型，所以现在越来越鸡肋了。 纯粹在内存中完成的排序比例。数据来源于v$sysstat statistics sorts (disk) 和 sorts (memory).



sorts (disk)	45	0.01	0.00
sorts (memory)	1,154,835	369.76	0.20
sorts (rows)	911,885,722	291,973.66	156.87

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

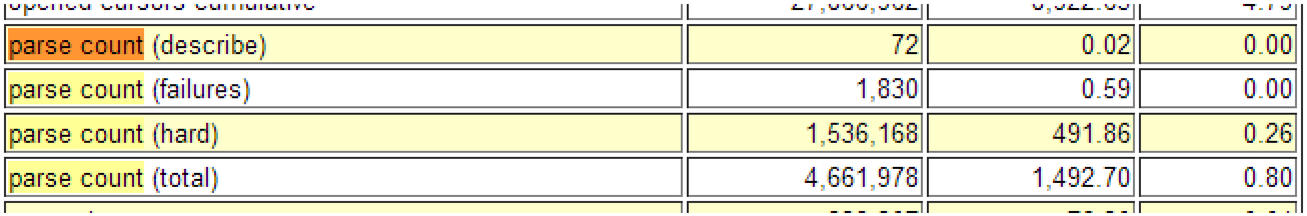


Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.99	Redo NoWait %:	99.98
Buffer Hit %:	98.46	In-memory Sort %:	100.00
Library Hit %:	95.54	Soft Parse %:	67.05
Execute to Parse %:	90.43	Latch Hit %:	91.46
Parse CPU to Parse Elapsd %:	2.34	% Non-Parse CPU:	93.62

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%

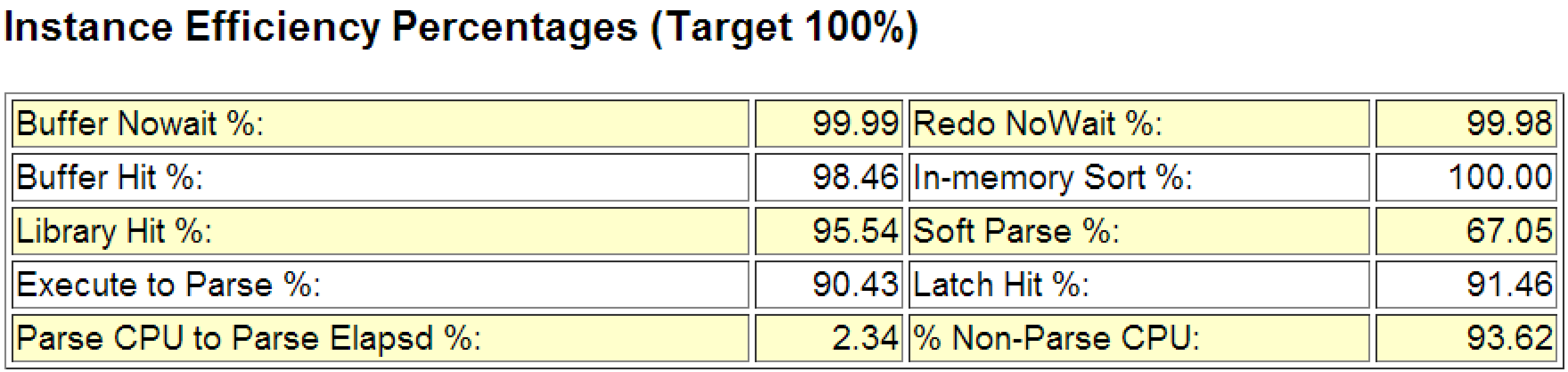
Soft Parse: 软解析比例，无需多说的经典指标，数据来源v$sysstat statistics的parse count(total)和parse count(hard)。 合理值>95%



opened cursors cumulative	[illegible]	[illegible]	[illegible]
parse count (describe)	72	0.02	0.00
parse count (failures)	1,830	0.59	0.00
parse count (hard)	1,536,168	491.86	0.26
parse count (total)	4,661,978	1,492.70	0.80

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



Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.99	Redo NoWait %:	99.98
Buffer Hit %:	98.46	In-memory Sort %:	100.00
Library Hit %:	95.54	Soft Parse %:	67.05
Execute to Parse %:	90.43	Latch Hit %:	91.46
Parse CPU to Parse Elapsd %:	2.34	% Non-Parse CPU:	93.62

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

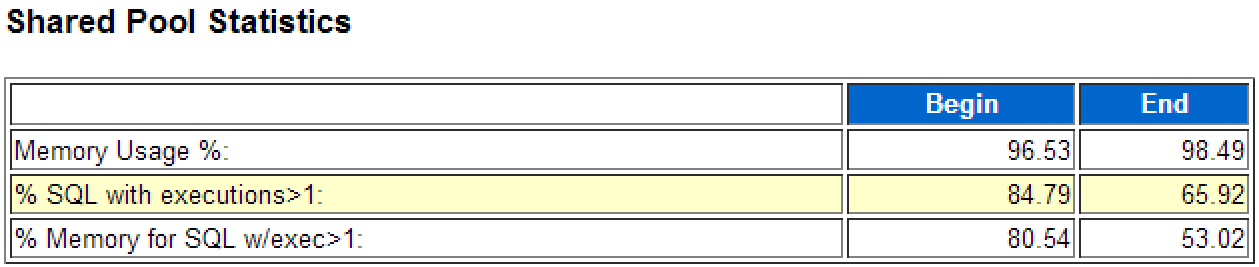
Parse CPU To Parse Elapsd:该指标反映了 快照内解析CPU时间和总的解析时间的比值(Parse CPU Time/ Parse Elapsed Time)； 若该指标水平很低，那么说明在整个解析过程中 实际在CPU上运算的时间是很短的，而主要的解析时间都耗费在各种其他非空闲的等待事件上了(如latch:shared pool,row cache lock之类等) 数据来源 V$sysstat 的 parse time cpu和parse time elapsed

%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

http://www.askmaclean.com/archives/know-more-about-awr-parse-statistics.html

Shared Pool Statistics

反应SQL重用率和共享池中cursor对内存的使用



Shared Pool Statistics

	Begin	End
Memory Usage %:	96.53	98.49
% SQL with executions>1:	84.79	65.92
% Memory for SQL w/exec>1:	80.54	53.02

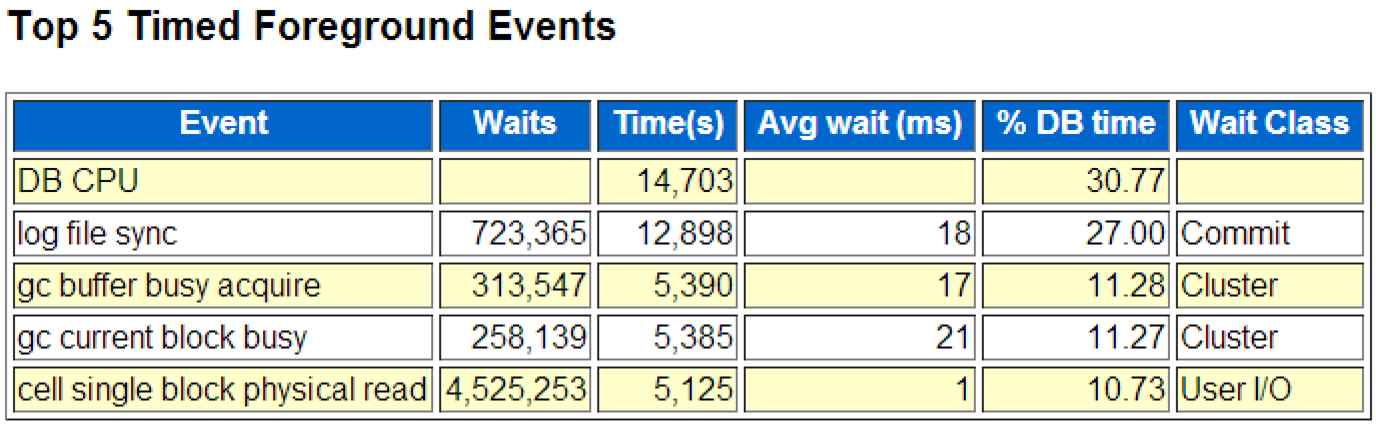
该环节提供一个大致的SQL重用及shared pool内存使用的评估。 应用是否共享SQL? 有多少内存是给只运行一次的SQL占掉的，对比共享SQL呢？

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→舞会主角

Top 5 万众瞩目，DBA为你倾倒！



Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		14,703		30.77	
log file sync	723,365	12,898	18	27.00	Commit
gc buffer busy acquire	313,547	5,390	17	11.28	Cluster
gc current block busy	258,139	5,385	21	11.27	Cluster
cell single block physical read	4,525,253	5,125	1	10.73	User I/O

基于Wait Interface的调优是目前的主流！每个指标都重要！

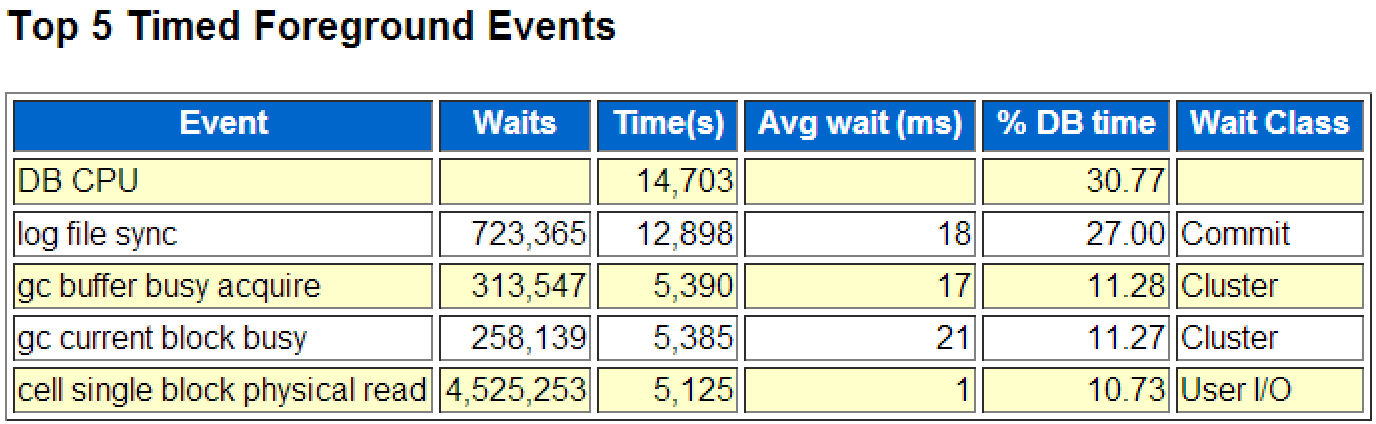
基于命中比例的调优，好比是统计局的报告， 张财主家财产100万，李木匠家财产1万， 平均财产50.5万。

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……

Top 5 Timed Foreground Events DB CPU/Cpu Time

Top 5 万众瞩目，DBA为你倾倒！



Top 5 Timed Foreground Events

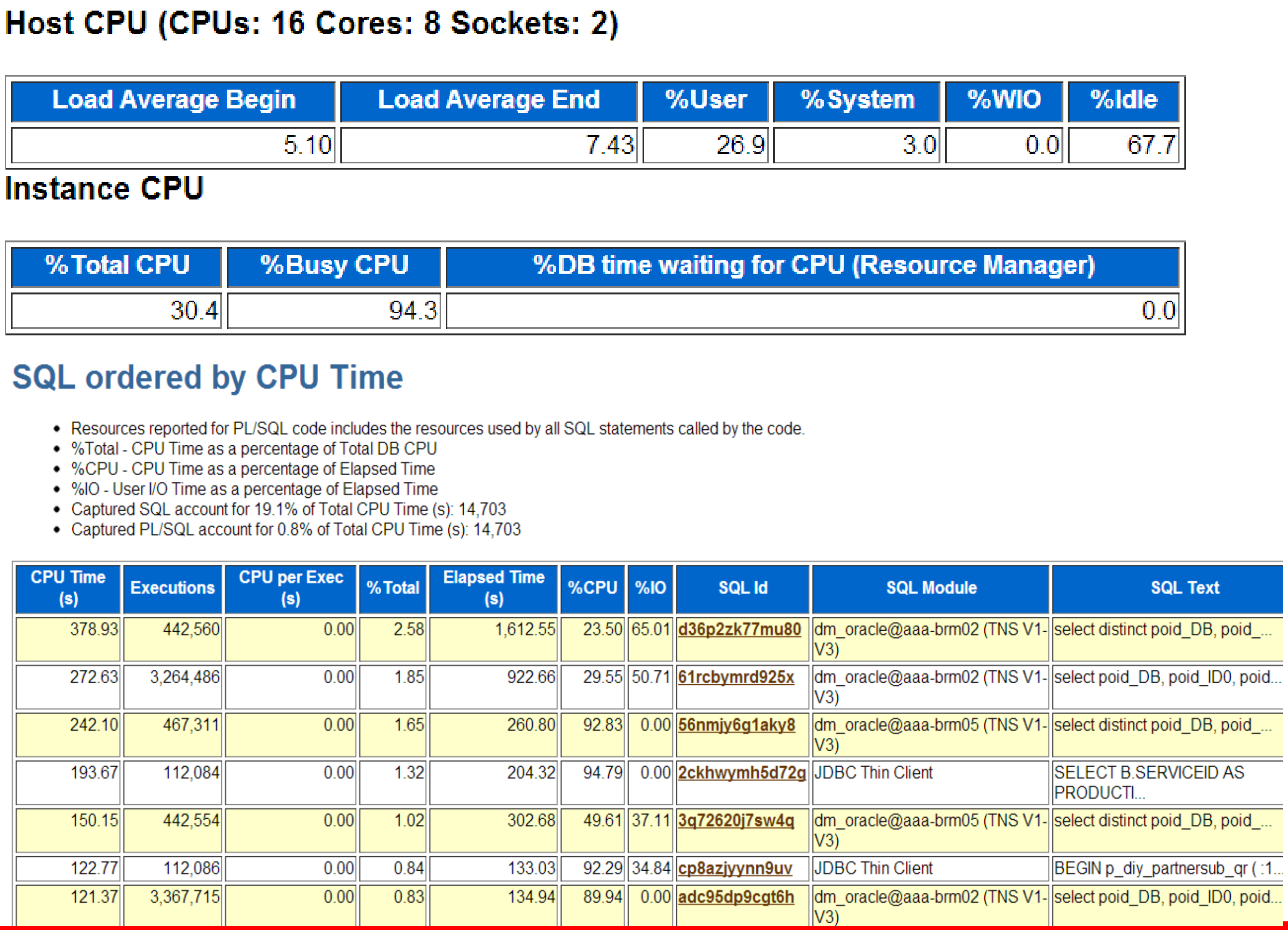
Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		14,703		30.77	
log file sync	723,365	12,898	18	27.00	Commit
gc buffer busy acquire	313,547	5,390	17	11.28	Cluster
gc current block busy	258,139	5,385	21	11.27	Cluster
cell single block physical read	4,525,253	5,125	1	10.73	User I/O

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！

Top 5 Timed Foreground Events DB CPU/Cpu Time

Top 5 万众瞩目，DBA为你倾倒！

结合Host CPU、Instance CPU、 SQL ordered by CPU Time一起看哦！



Host CPU (CPUs: 16 Cores: 8 Sockets: 2)

Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
5.10	7.43	26.9	3.0	0.0	67.7

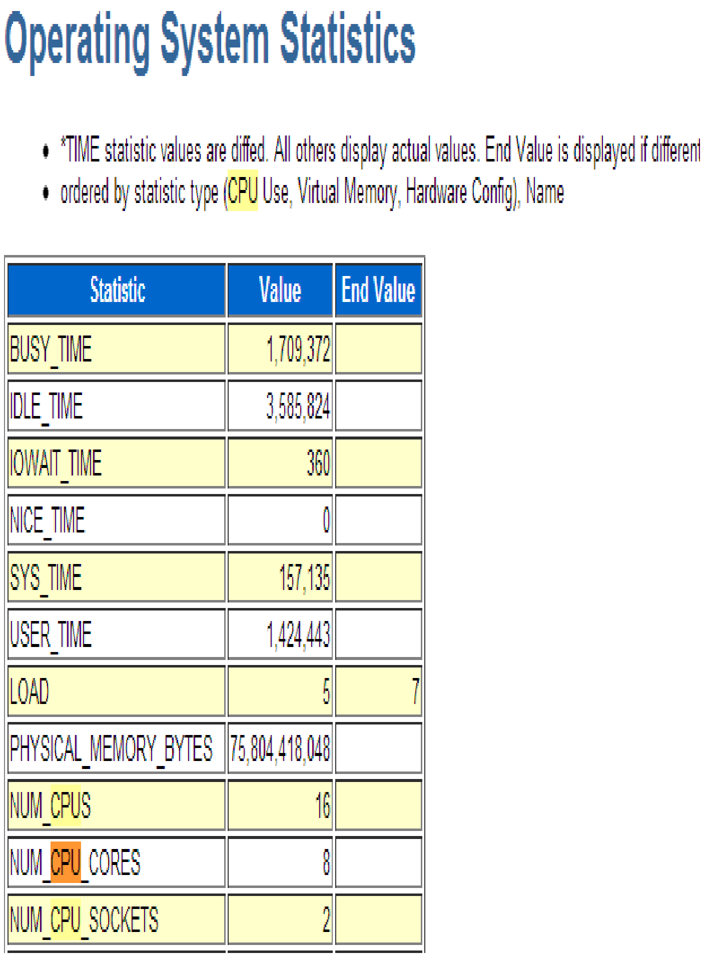
Instance CPU

%Total CPU	%Busy CPU	%DB time waiting for CPU (Resource Manager)
30.4	94.3	0.0

SQL ordered by CPU Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- %Total - CPU Time as a percentage of Total DB CPU
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 19.1% of Total CPU Time (s): 14,703
- Captured PL/SQL account for 0.8% of Total CPU Time (s): 14,703

CPU Time (s)	Executions	CPU per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	SQL Text
378.93	442,560	0.00	2.58	1,612.55	23.50	65.01	d36p2zk77mu80	dm_oracle@aaa-brm02 (TNS V1-V3)	select distinct poid_DB, poid_...
272.63	3,264,486	0.00	1.85	922.66	29.55	50.71	61rcbymrd925x	dm_oracle@aaa-brm02 (TNS V1-V3)	select poid_DB, poid_ID0, poid...
242.10	467,311	0.00	1.65	260.80	92.83	0.00	56nmjy6g1aky8	dm_oracle@aaa-brm05 (TNS V1-V3)	select distinct poid_DB, poid_...
193.67	112,084	0.00	1.32	204.32	94.79	0.00	2ckhwymh5d72g	JDBC Thin Client	SELECT B.SERVICEID AS PRODUCTI...
150.15	442,554	0.00	1.02	302.68	49.61	37.11	3q72620j7sw4q	dm_oracle@aaa-brm05 (TNS V1-V3)	select distinct poid_DB, poid_...
122.77	112,086	0.00	0.84	133.03	92.29	34.84	cp8azjyynn9uv	JDBC Thin Client	BEGIN p_diy_partnersub_qr (:1...
121.37	3,367,715	0.00	0.83	134.94	89.94	0.00	adc95dp9cgt6h	dm_oracle@aaa-brm02 (TNS V1-V3)	select poid_DB, poid_ID0, poid...



Operating System Statistics

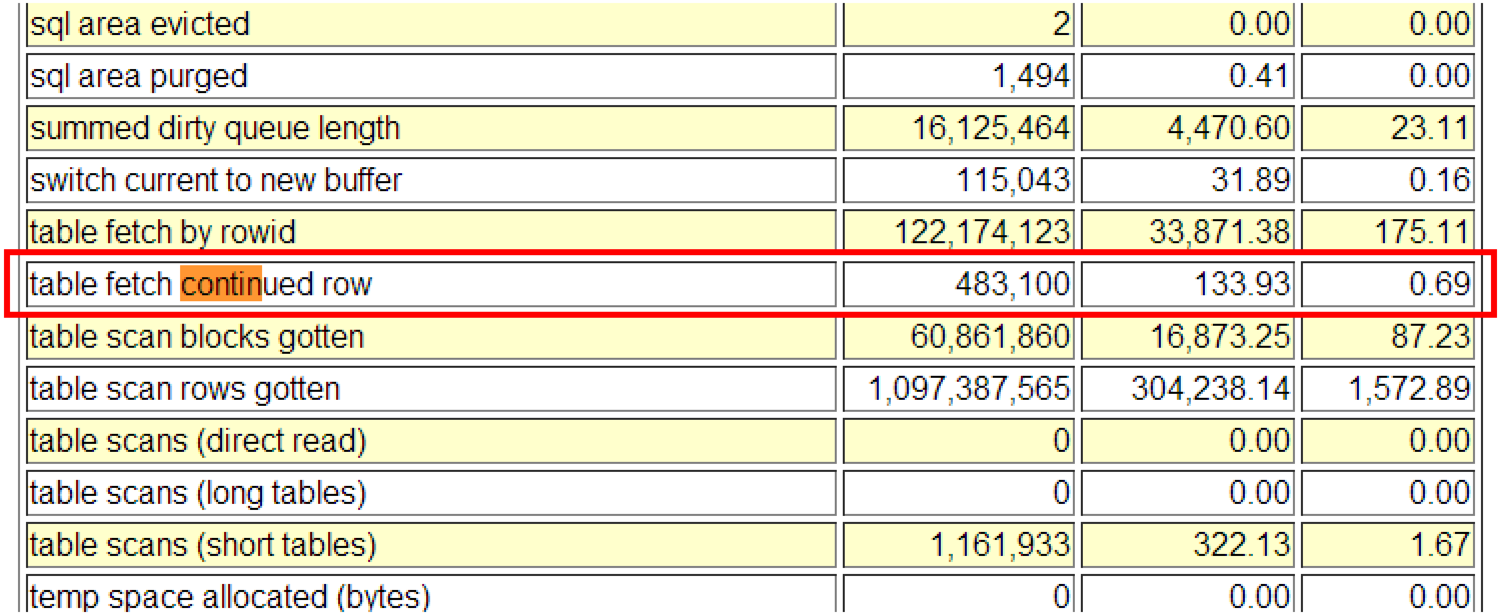
- *TIME statistic values are diffed. All others display actual values. End Value is displayed if different
- ordered by statistic type (CPU Use, Virtual Memory, Hardware Config), Name

Statistic	Value	End Value
BUSY_TIME	1,709,372	
IDLE_TIME	3,585,824	
IOWAIT_TIME	360	
NICE_TIME	0	
SYS_TIME	157,135	
USER_TIME	1,424,443	
LOAD	5	7
PHYSICAL_MEMORY_BYTES	75,804,418,048	
NUM_CPUS	16	
NUM_CPU_CORES	8	
NUM_CPU_SOCKETS	2	

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

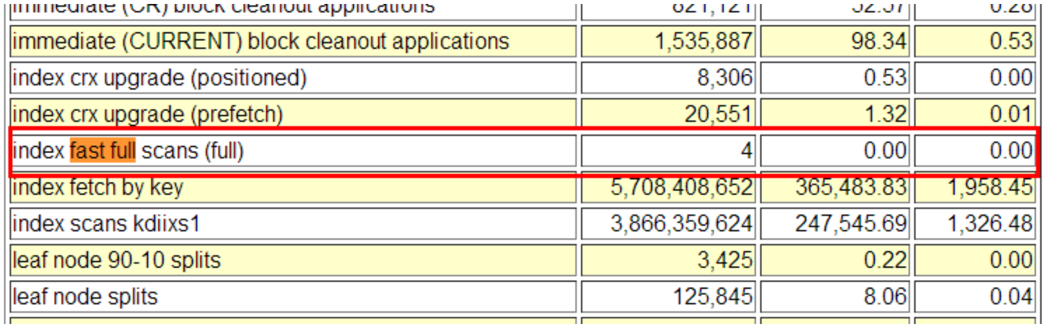


sql area evicted	2	0.00	0.00
sql area purged	1,494	0.41	0.00
summed dirty queue length	16,125,464	4,470.60	23.11
switch current to new buffer	115,043	31.89	0.16
table fetch by rowid	122,174,123	33,871.38	175.11
table fetch continued row	483,100	133.93	0.69
table scan blocks gotten	60,861,860	16,873.25	87.23
table scan rows gotten	1,097,387,565	304,238.14	1,572.89
table scans (direct read)	0	0.00	0.00
table scans (long tables)	0	0.00	0.00
table scans (short tables)	1,161,933	322.13	1.67
temp space allocated (bytes)	0	0.00	0.00

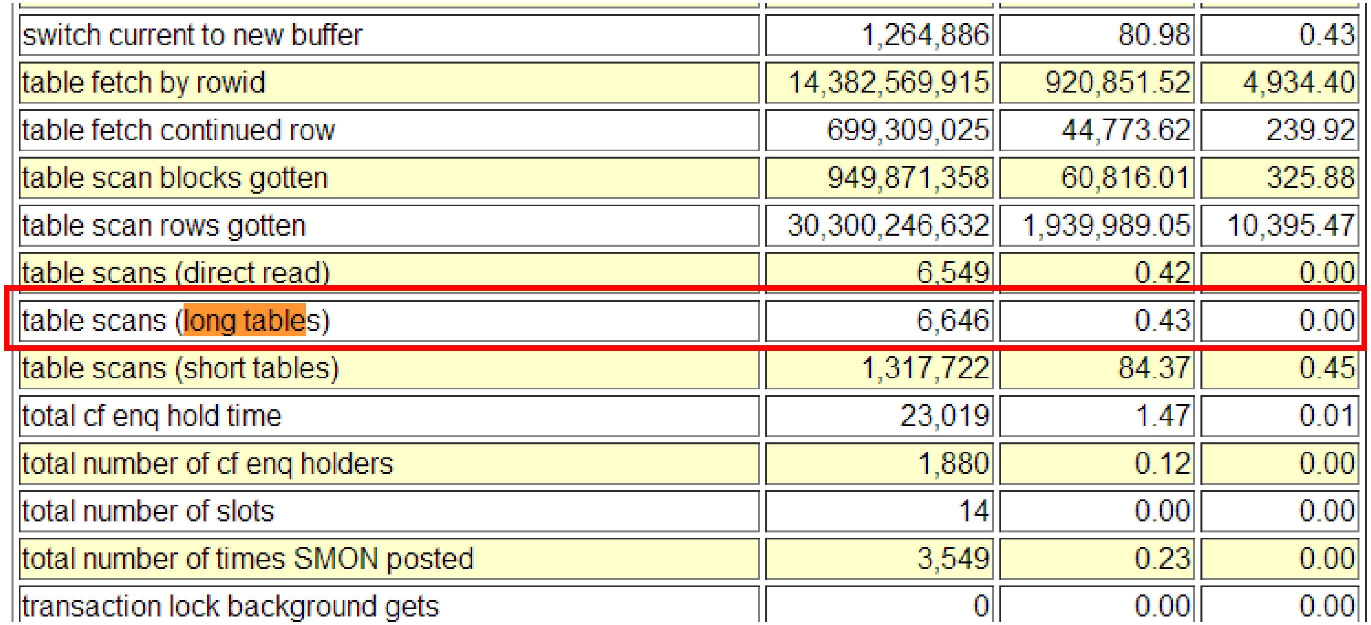
db file scattered read - Top event

Avg wait time应当小于20ms

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

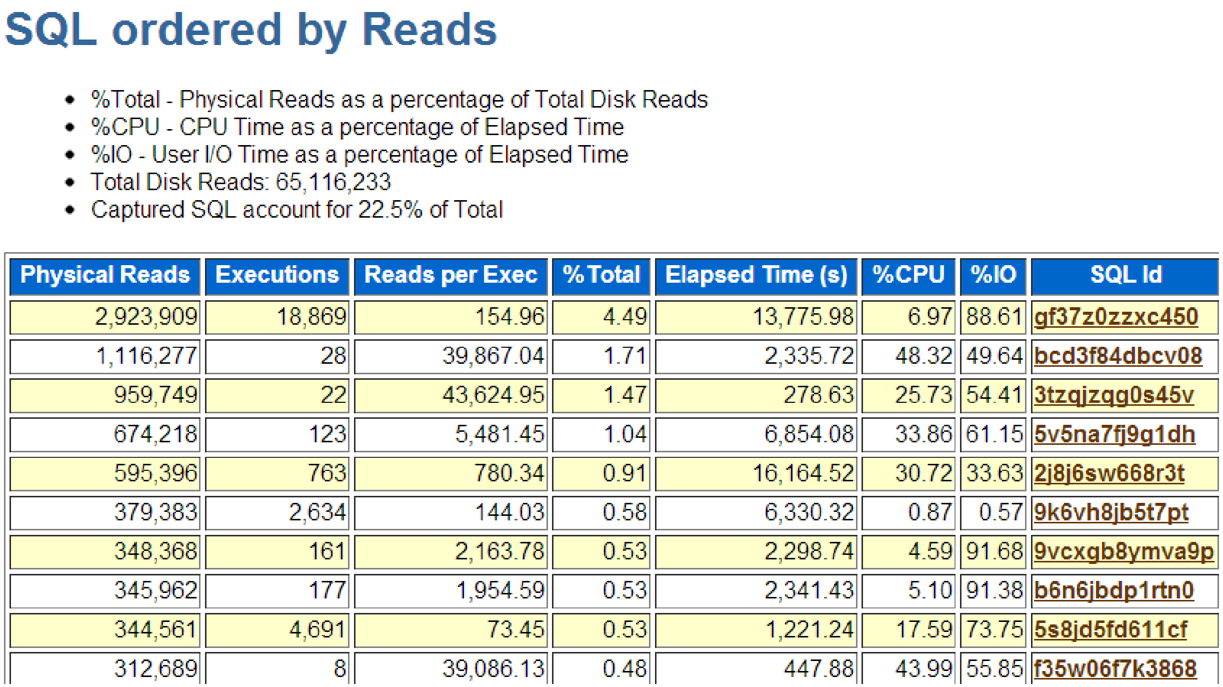


immediate (CR) block cleanout applications	821,121	52.57	0.28
immediate (CURRENT) block cleanout applications	1,535,887	98.34	0.53
index crx upgrade (positioned)	8,306	0.53	0.00
index crx upgrade (prefetch)	20,551	1.32	0.01
index fast full scans (full)	4	0.00	0.00
index fetch by key	5,708,408,652	365,483.83	1,958.45
index scans kdiixs1	3,866,359,624	247,545.69	1,326.48
leaf node 90-10 splits	3,425	0.22	0.00
leaf node splits	125,845	8.06	0.04



switch current to new buffer	1,264,886	80.98	0.43
table fetch by rowid	14,382,569,915	920,851.52	4,934.40
table fetch continued row	699,309,025	44,773.62	239.92
table scan blocks gotten	949,871,358	60,816.01	325.88
table scan rows gotten	30,300,246,632	1,939,989.05	10,395.47
table scans (direct read)	6,549	0.42	0.00
table scans (long tables)	6,646	0.43	0.00
table scans (short tables)	1,317,722	84.37	0.45
total cf enq hold time	23,019	1.47	0.01
total number of cf enq holders	1,880	0.12	0.00
total number of slots	14	0.00	0.00
total number of times SMON posted	3,549	0.23	0.00
transaction lock background gets	0	0.00	0.00

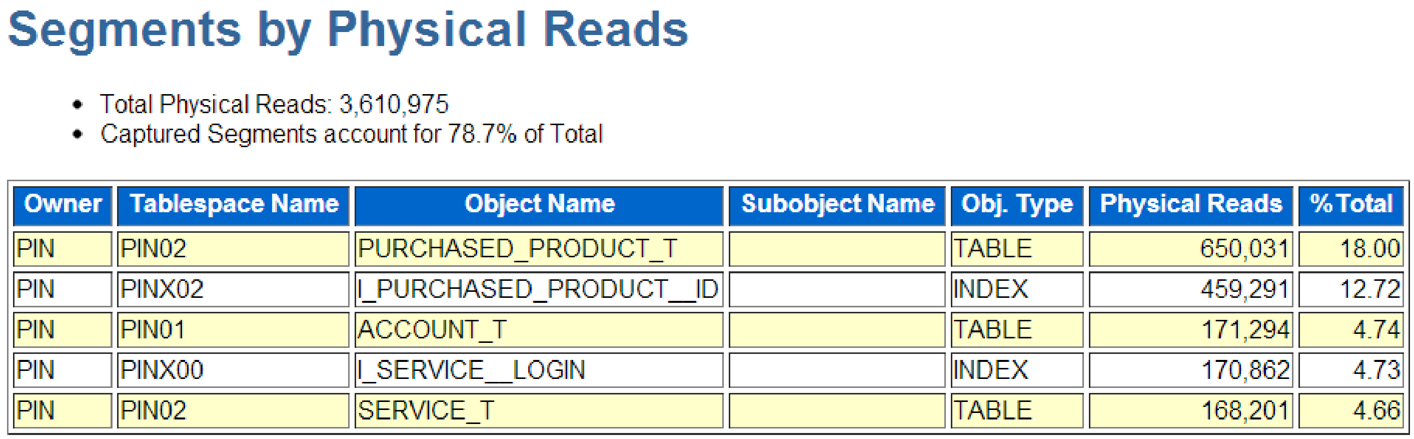
db file sequential/scattered read - Top event



SQL ordered by Reads

- %Total - Physical Reads as a percentage of Total Disk Reads
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Total Disk Reads: 65,116,233
- Captured SQL account for 22.5% of Total

Physical Reads	Executions	Reads per Exec	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id
2,923,909	18,869	154.96	4.49	13,775.98	6.97	88.61	gf37z0zzxc450
1,116,277	28	39,867.04	1.71	2,335.72	48.32	49.64	bcd3f84dbcv08
959,749	22	43,624.95	1.47	278.63	25.73	54.41	3tzqjzqg0s45v
674,218	123	5,481.45	1.04	6,854.08	33.86	61.15	5v5na7fj9g1dh
595,396	763	780.34	0.91	16,164.52	30.72	33.63	2j8j6sw668r3t
379,383	2,634	144.03	0.58	6,330.32	0.87	0.57	9k6vh8jb5t7pt
348,368	161	2,163.78	0.53	2,298.74	4.59	91.68	9vcxgb8ymva9p
345,962	177	1,954.59	0.53	2,341.43	5.10	91.38	b6n6jbdp1rtn0
344,561	4,691	73.45	0.53	1,221.24	17.59	73.75	5s8jd5fd611cf
312,689	8	39,086.13	0.48	447.88	43.99	55.85	f35w06f7k3868



Segments by Physical Reads

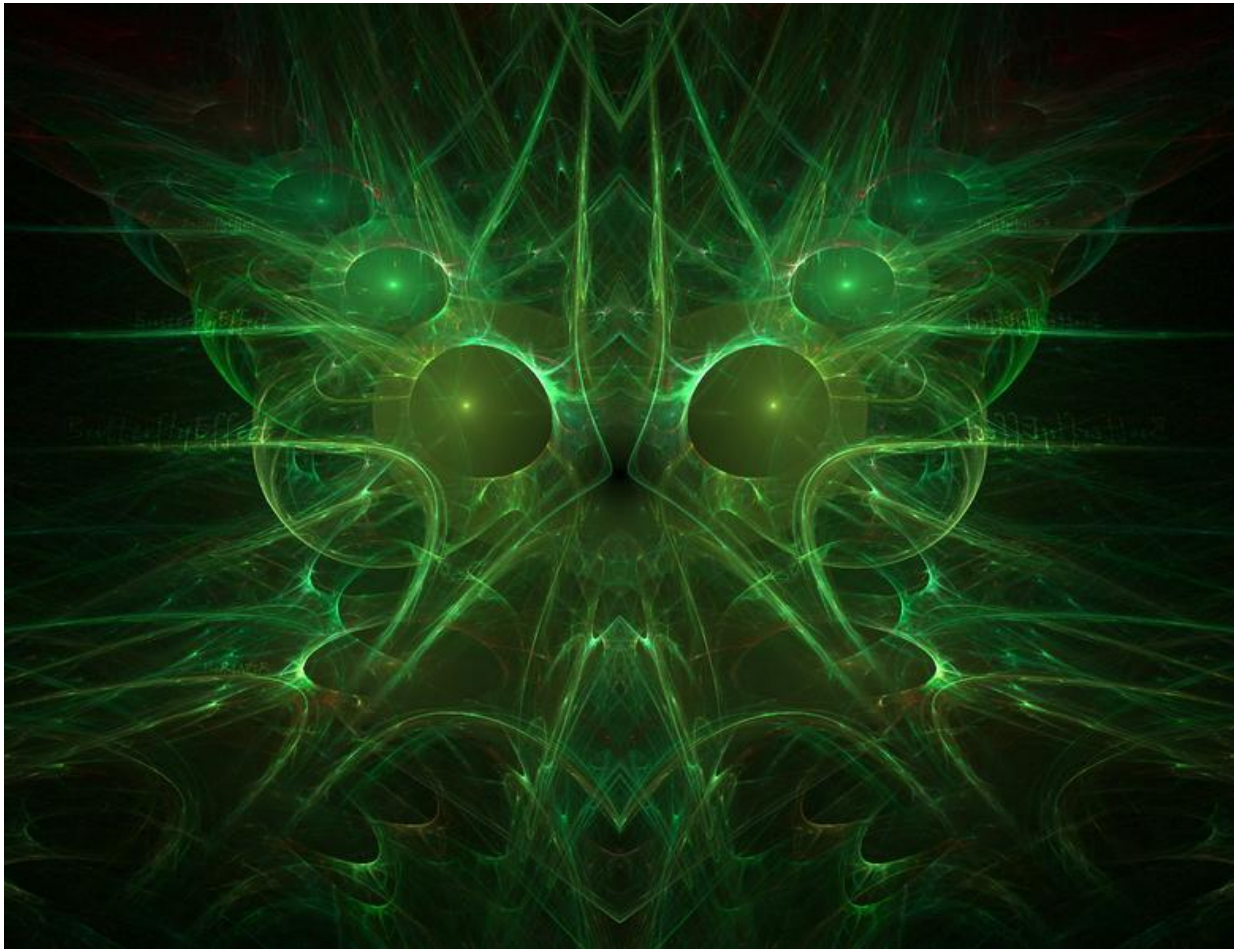
- Total Physical Reads: 3,610,975
- Captured Segments account for 78.7% of Total

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Physical Reads	%Total
PIN	PIN02	PURCHASED_PRODUCT_T		TABLE	650,031	18.00
PIN	PINX02	I_PURCHASED_PRODUCT__ID		INDEX	459,291	12.72
PIN	PIN01	ACCOUNT_T		TABLE	171,294	4.74
PIN	PINX00	I_SERVICE__LOGIN		INDEX	170,862	4.73
PIN	PIN02	SERVICE_T		TABLE	168,201	4.66

Log file sync蝴蝶效应

Log file sync ➔ enq: TX， gc buffer busy，buffer busy wait

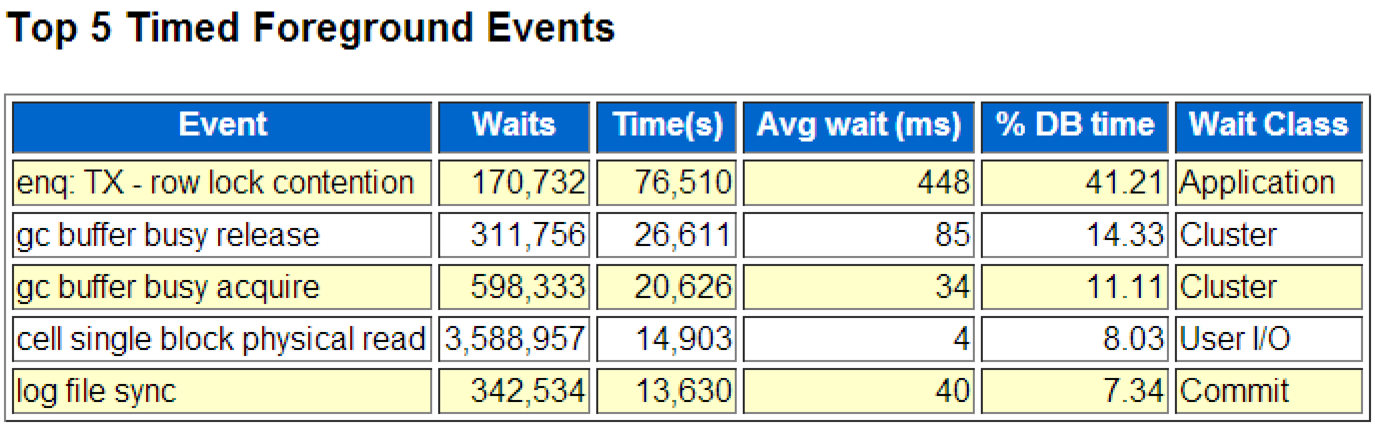
等待事件的混沌理论，性能不是线性的，而是多纬度的



Log file sync实战案例

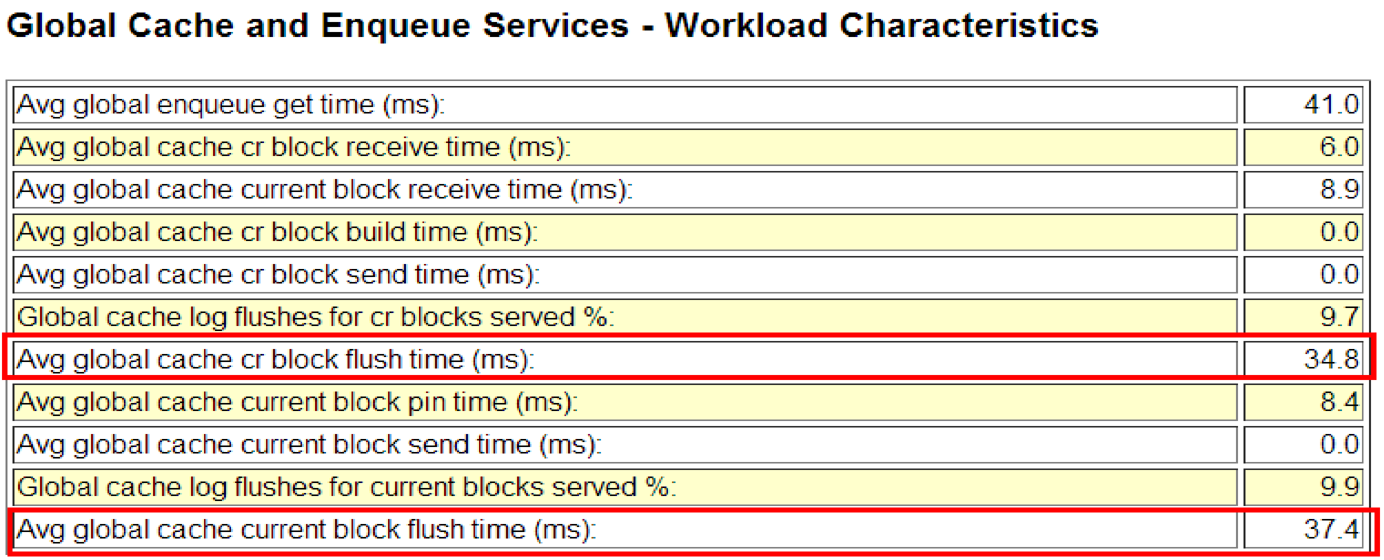
Log file sync ➔ enq: TX， gc buffer busy，buffer busy wait , enq:TX index contention

等待事件的混沌理论，性能不是线性的，而是多纬度的



Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
enq: TX - row lock contention	170,732	76,510	448	41.21	Application
gc buffer busy release	311,756	26,611	85	14.33	Cluster
gc buffer busy acquire	598,333	20,626	34	11.11	Cluster
cell single block physical read	3,588,957	14,903	4	8.03	User I/O
log file sync	342,534	13,630	40	7.34	Commit

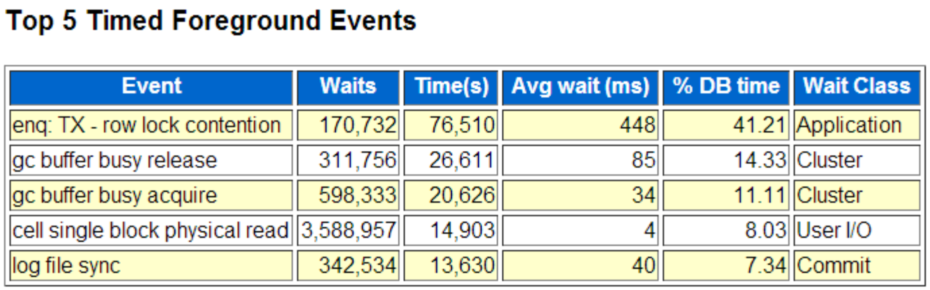


Global Cache and Enqueue Services - Workload Characteristics

Avg global enqueue get time (ms):	41.0
Avg global cache cr block receive time (ms):	6.0
Avg global cache current block receive time (ms):	8.9
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 %:	9.7
Avg global cache cr block flush time (ms):	34.8
Avg global cache current block pin time (ms):	8.4
Avg global cache current block send time (ms):	0.0
Global cache log flushes for current blocks served %:	9.9
Avg global cache current block flush time (ms):	37.4

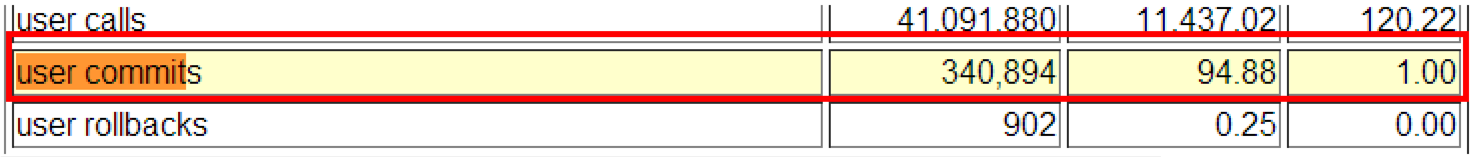
Log file sync实战案例

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➔ enq:TX 大幅出现

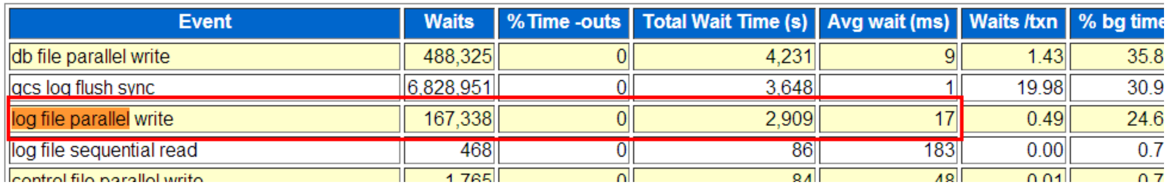


Top 5 Timed Foreground Events

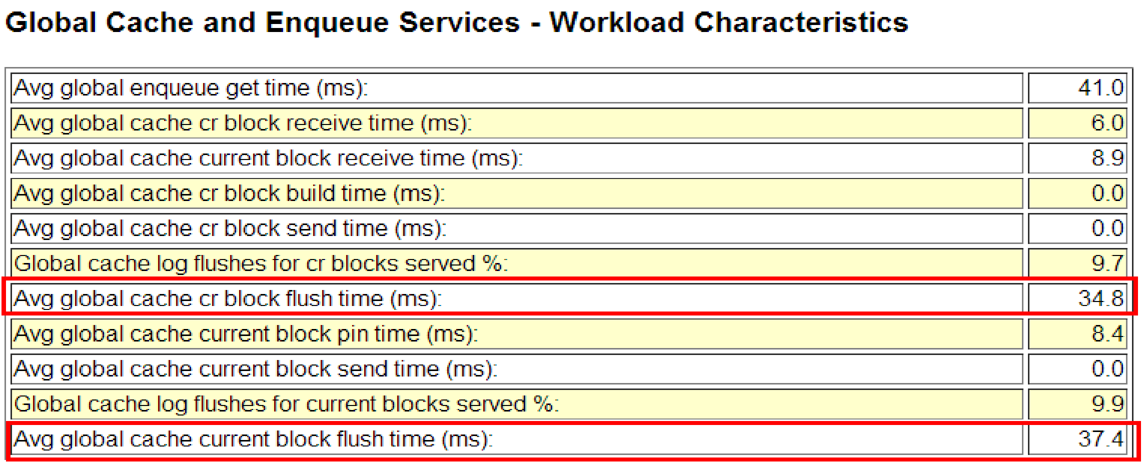
Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
enq: TX - row lock contention	170,732	76,510	448	41.21	Application
gc buffer busy release	311,756	26,611	85	14.33	Cluster
gc buffer busy acquire	598,333	20,626	34	11.11	Cluster
cell single block physical read	3,588,957	14,903	4	8.03	User I/O
log file sync	342,534	13,630	40	7.34	Commit



user calls	41,091,880	11,437.02	120.22
user commits	340,894	94.88	1.00
user rollbacks	902	0.25	0.00



Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn	% bg time
db file parallel write	488,325	0	4,231	9	1.43	35.8
gcs log flush sync	6,828,951	0	3,648	1	19.98	30.9
log file parallel write	167,338	0	2,909	17	0.49	24.6
log file sequential read	468	0	86	183	0.00	0.7
control file parallel write	1,765	0	84	48	0.01	0.7



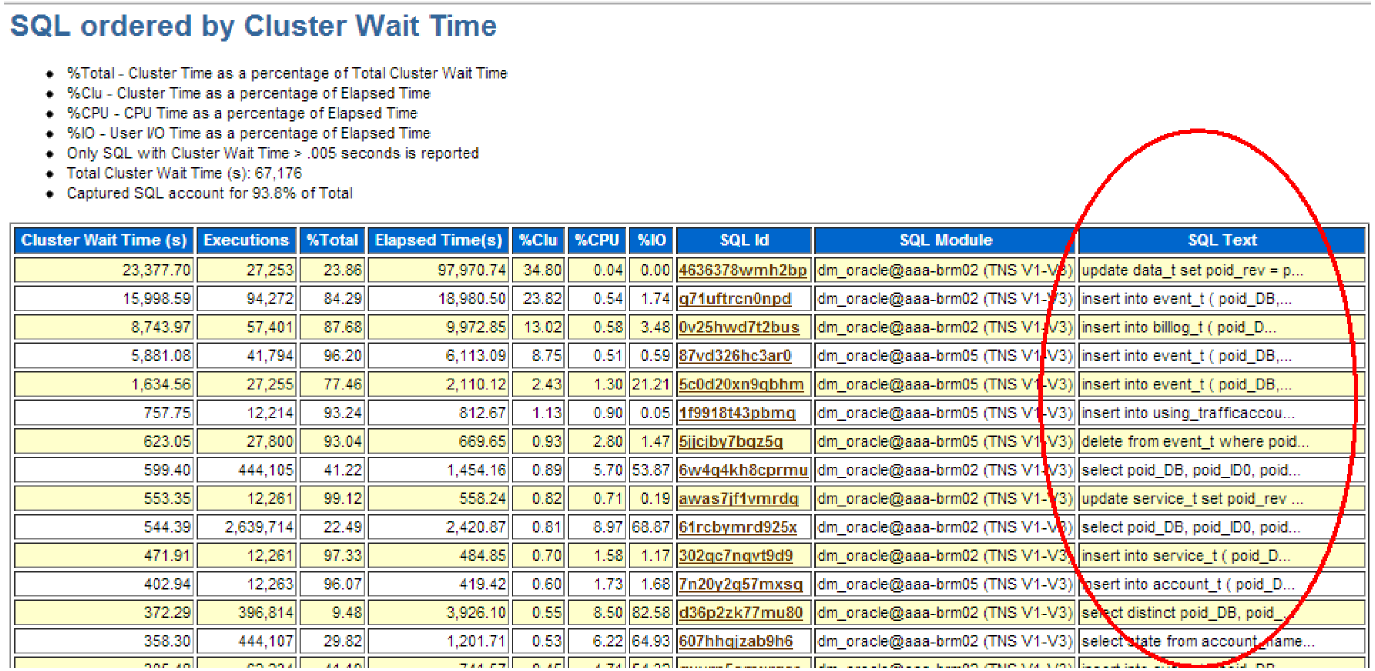
Global Cache and Enqueue Services - Workload Characteristics

Avg global enqueue get time (ms):	41.0
Avg global cache cr block receive time (ms):	6.0
Avg global cache current block receive time (ms):	8.9
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 %:	9.7
Avg global cache cr block flush time (ms):	34.8
Avg global cache current block pin time (ms):	8.4
Avg global cache current block send time (ms):	0.0
Global cache log flushes for current blocks served %:	9.9
Avg global cache current block flush time (ms):	37.4

Log file sync实战案例

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

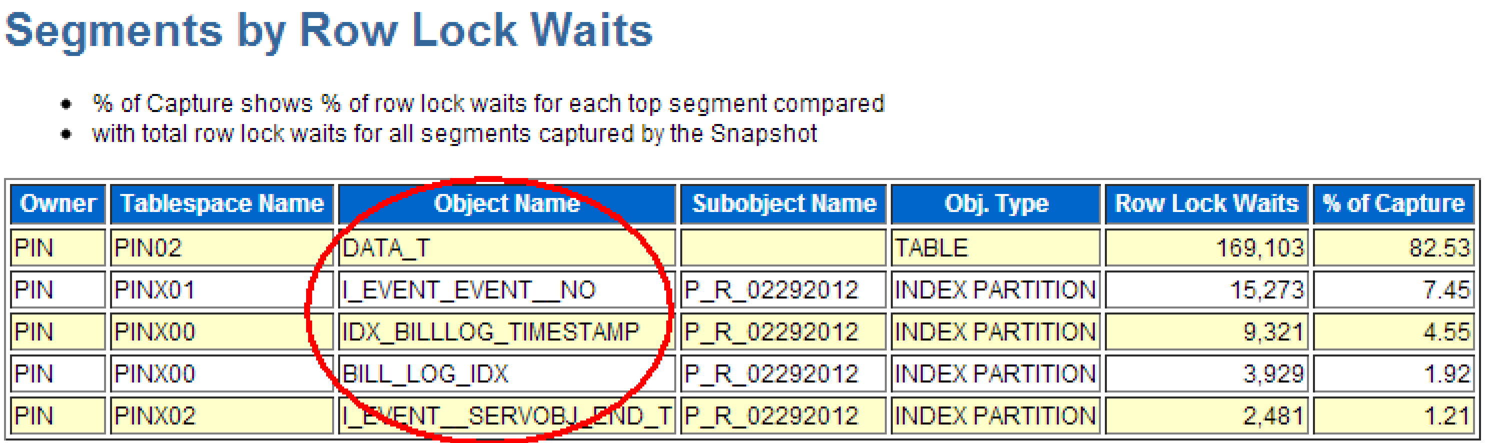
最主要是update和insert 受影响，前台处理业务速度放慢。方向对了就处处对得上了



SQL ordered by Cluster Wait Time

- %Total - Cluster Time as a percentage of Total Cluster Wait Time
- %Clu - Cluster Time as a percentage of Elapsed Time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Only SQL with Cluster Wait Time > .005 seconds is reported
- Total Cluster Wait Time (s): 67,176
- Captured SQL account for 93.8% of Total

Cluster Wait Time (s)	Executions	%Total	Elapsed Time(s)	%Clu	%CPU	%IO	SQL Id	SQL Module	SQL Text
23,377.70	27,253	23.86	97,970.74	34.80	0.04	0.00	4636378wmh2bp	dm_oracle@aaa-brm02 (TNS V1-V3)	update data_t set poid_rev = p...
15,998.59	94,272	84.29	18,980.50	23.82	0.54	1.74	g71uftrcn0npd	dm_oracle@aaa-brm02 (TNS V1-V3)	insert into event_t (poid_DB,...
8,743.97	57,401	87.68	9,972.85	13.02	0.58	3.48	0v25hwd7t2bus	dm_oracle@aaa-brm02 (TNS V1-V3)	insert into billlog_t (poid_D...
5,881.08	41,794	96.20	6,113.09	8.75	0.51	0.59	87vd326hc3ar0	dm_oracle@aaa-brm05 (TNS V1-V3)	insert into event_t (poid_DB,...
1,634.56	27,255	77.46	2,110.12	2.43	1.30	21.21	5c0d20xn9qbhm	dm_oracle@aaa-brm05 (TNS V1-V3)	insert into event_t (poid_DB,...
757.75	12,214	93.24	812.67	1.13	0.90	0.05	1f9918t43pbmq	dm_oracle@aaa-brm05 (TNS V1-V3)	insert into using_trafficaccou...
623.05	27,800	93.04	669.65	0.93	2.80	1.47	5jjcjby7bqz5q	dm_oracle@aaa-brm05 (TNS V1-V3)	delete from event_t where poid...
599.40	444,105	41.22	1,454.16	0.89	5.70	53.87	6w4q4kh8cprmu	dm_oracle@aaa-brm02 (TNS V1-V3)	select poid_DB, poid_ID0, poid...
553.35	12,261	99.12	558.24	0.82	0.71	0.19	awas7jf1vmrdq	dm_oracle@aaa-brm02 (TNS V1-V3)	update service_t set poid_rev ...
544.39	2,639,714	22.49	2,420.87	0.81	8.97	68.87	61rcbymrd925x	dm_oracle@aaa-brm02 (TNS V1-V3)	select poid_DB, poid_ID0, poid...
471.91	12,261	97.33	484.85	0.70	1.58	1.17	302qc7nqvt9d9	dm_oracle@aaa-brm02 (TNS V1-V3)	insert into service_t (poid_D...
402.94	12,263	96.07	419.42	0.60	1.73	1.68	7n20y2q57mxsq	dm_oracle@aaa-brm05 (TNS V1-V3)	insert into account_t (poid_D...
372.29	396,814	9.48	3,926.10	0.55	8.50	82.58	d36p2zk77mu80	dm_oracle@aaa-brm02 (TNS V1-V3)	select distinct poid_DB, poid_...
358.30	444,107	29.82	1,201.71	0.53	6.22	64.93	607hhqjzab9h6	dm_oracle@aaa-brm02 (TNS V1-V3)	select state from account_name...



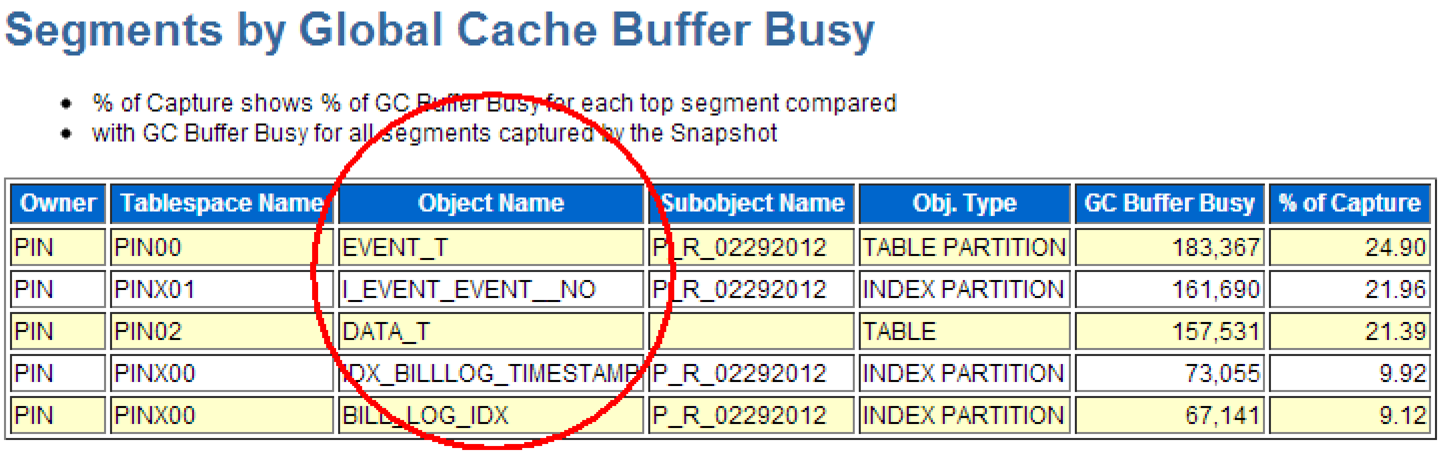
Segments by Row Lock Waits

- % of Capture shows % of row lock waits for each top segment compared
- with total row lock waits for all segments captured by the Snapshot

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Row Lock Waits	% of Capture
PIN	PIN02	DATA_T		TABLE	169,103	82.53
PIN	PINX01	I_EVENT_EVENT__NO	P_R_02292012	INDEX PARTITION	15,273	7.45
PIN	PINX00	IDX_BILLLOG_TIMESTAMP	P_R_02292012	INDEX PARTITION	9,321	4.55
PIN	PINX00	BILL_LOG_IDX	P_R_02292012	INDEX PARTITION	3,929	1.92
PIN	PINX02	I_EVENT__SERVOBJ_END_T	P_R_02292012	INDEX PARTITION	2,481	1.21

Log file sync实战案例

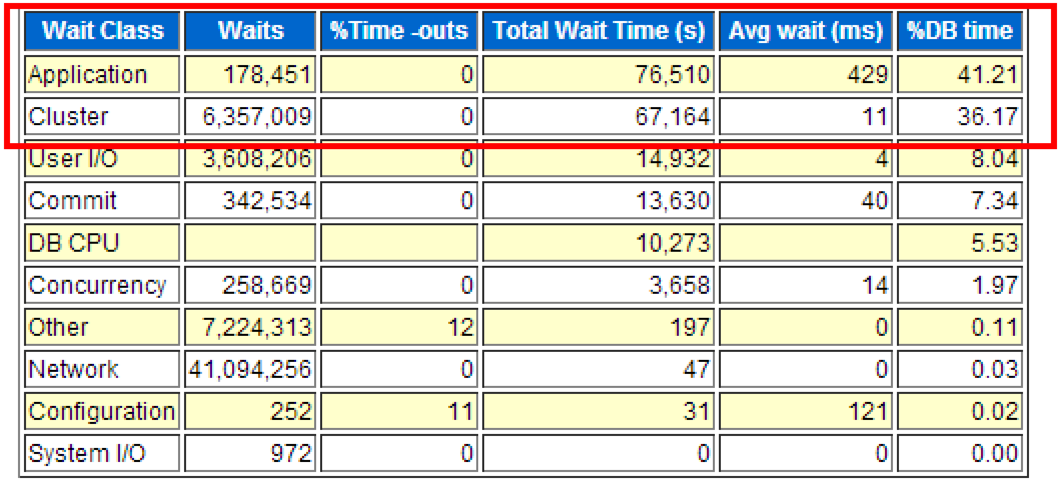
Global Buffer Busy ➔ gc buffer busy acquire/release 受影响的segment



Segments by Global Cache Buffer Busy

- % of Capture shows % of GC Buffer Busy for each top segment compared
- with GC Buffer Busy for all segments captured by the Snapshot

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	GC Buffer Busy	% of Capture
PIN	PIN00	EVENT_T	P_R_02292012	TABLE PARTITION	183,367	24.90
PIN	PINX01	I_EVENT_EVENT__NO	P_R_02292012	INDEX PARTITION	161,690	21.96
PIN	PIN02	DATA_T		TABLE	157,531	21.39
PIN	PINX00	IDX_BILLLOG_TIMESTAMP	P_R_02292012	INDEX PARTITION	73,055	9.92
PIN	PINX00	BILL_LOG_IDX	P_R_02292012	INDEX PARTITION	67,141	9.12

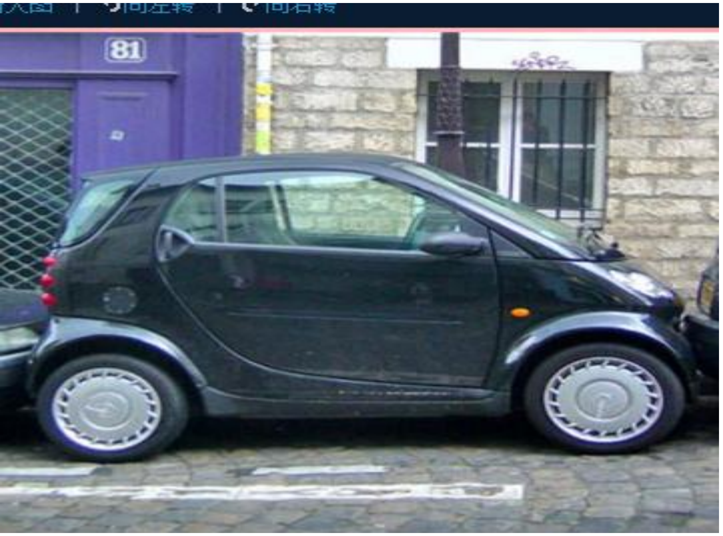


Wait Class	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	%DB time
Application	178,451	0	76,510	429	41.21
Cluster	6,357,009	0	67,164	11	36.17
User I/O	3,608,206	0	14,932	4	8.04
Commit	342,534	0	13,630	40	7.34
DB CPU			10,273		5.53
Concurrency	258,669	0	3,658	14	1.97
Other	7,224,313	12	197	0	0.11
Network	41,094,256	0	47	0	0.03
Configuration	252	11	31	121	0.02
System I/O	972	0	0	0	0.00

性能优化的多维度理论

- 增加了cpu➔更大的并发量，更多的并发争用
- 调整了Io存储➔ 更少的IO，更多的CPU计算，更高的cpu使用率
- Redo写得慢➔ 影响commit，造成enq:tx和gc buffer busy等待等
- Datafile写得慢➔ 检查点完不成，日志无法切换，前台DML hang
- Sequence nocache➔ 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➔ 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



经过Maclean

一番打造

但是即便再

打造也不可

能。。。。

才开了个头哦。。。

To Be Continued……………………

つづく………………

敬请期待开Oracle调优鹰眼，深入理解AWR性能报告第二讲

更多信息

www.askmaclean.com

tuning 🔍

or http://www.askmaclean.com/archives/tag/tuning

Question & Answer

If you have more questions later, feel free to ask