ACOUG
All China Oracle User Group
中国 Oracle 用户组

YUNHE
YUNHE ENMO (BEIJING) TECHNOLOGY CO.,LTD

Oracle数据库架构演进和性能优化实践

盖国强（Eygle）
电话:13911812803
邮件:eygle@enmotech.com
微博:weibo.com/eygle

SACC2012

YUNHE

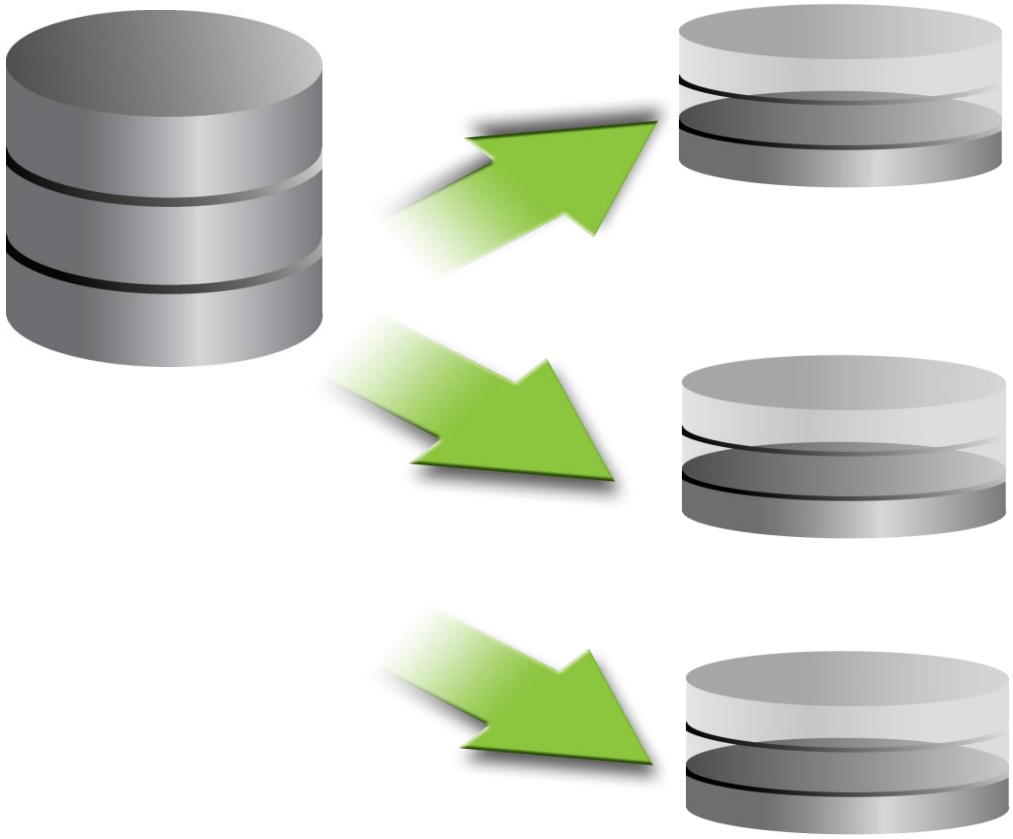
Who am *I*

- 盖国强　云和恩墨（北京）信息技术有限公司 创始人
  - 盖国强是国内第一个Oracle ACE及ACE总监
  - 有超过10年的Oracle从业经验

    至今仍然奋战在技术前线
  - 国内最大数据库技术论坛ITPUB的主要倡导者之一，致力于技术分享与传播，截至2011年已经出版了10本技术书籍，自2010年开始，主编出版《Oracle DBA手记》系列书籍
  - 2010年，他和张乐奕共同创建了旨在开展技术交流的中国Oracle用户组（ACOUG – All China Oracle User Group），并开展了持续的公益活动


SACC2012

数据库架构的演进-合久必分



企业的经历：

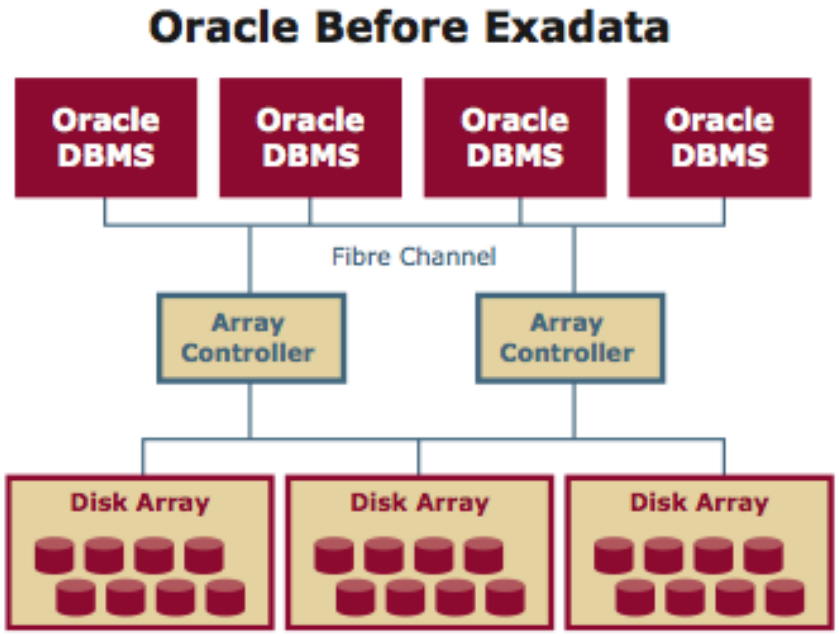
- 数据累积 性能衰减
- 拆分数据表
- 分割数据库
- 分布式数据库 - 去O
- 异构与迁移 - 去IE

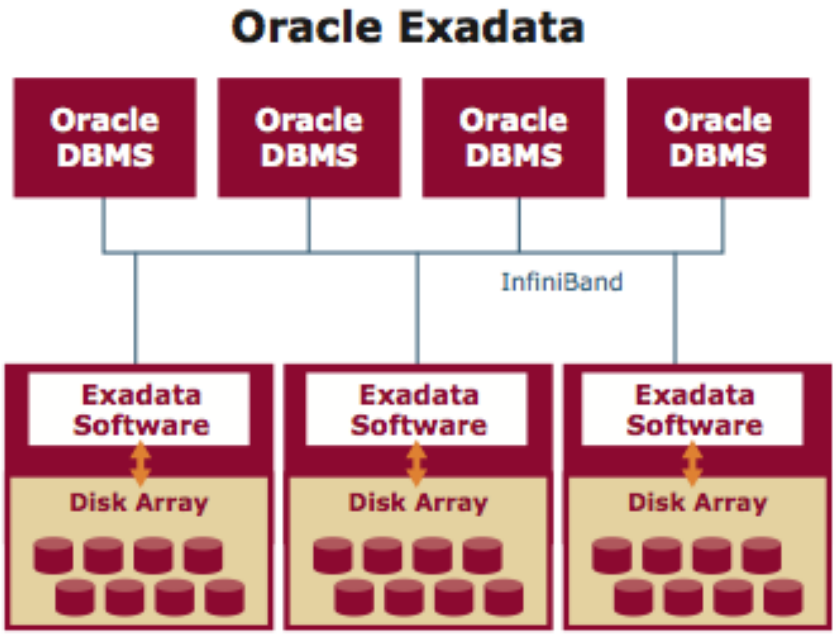
企业的目标：

- 提升性能
- 提高稳定性
- 保障数据安全
- 降低TCO

YUNHE

数据库架构的演进-合久必分

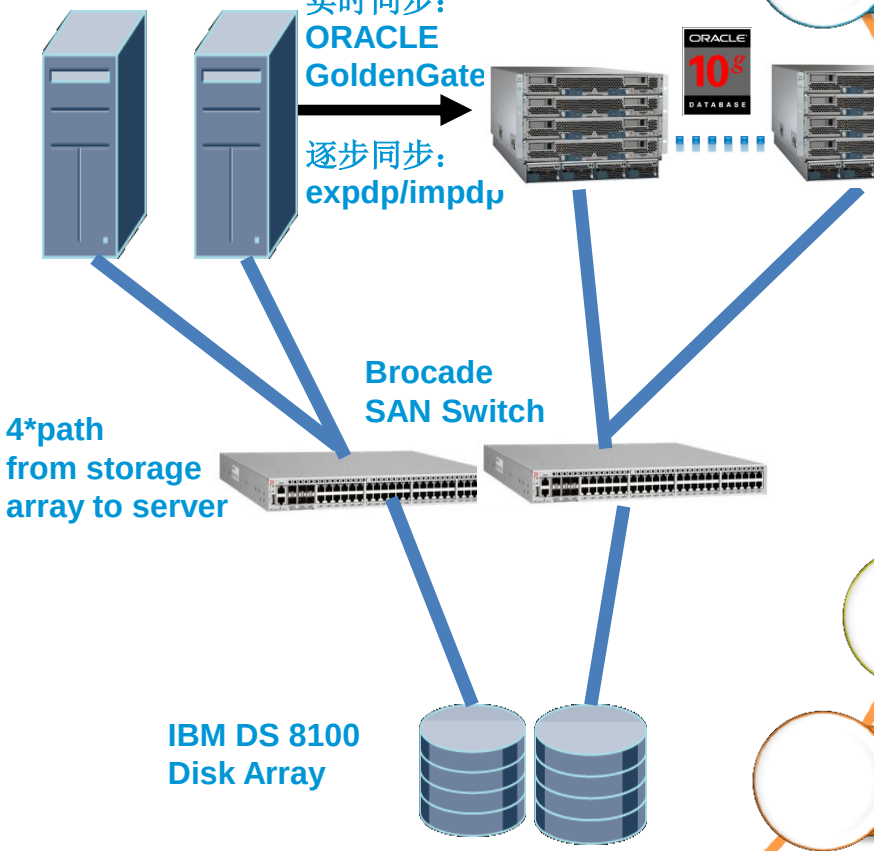




SACC2012

YUNHE

数据库架构的演进-开放平台整合迁移



本项目为广西公安厅信息科警务综合数据库Oracle RAC迁移

警务综合共计59个业务系统，约9T数据

迁移后警务综合平台将运行在4个Active UCS（2个Standby）上运行

2011年12月，AS参与迁移规划

2012年3月，某Partner为公安厅做迁移，没有成功，无法在规定窗口完成迁移

2012年4月，云和恩墨介入在4小时内完成数据迁移

数据库架构的演进-分久必合

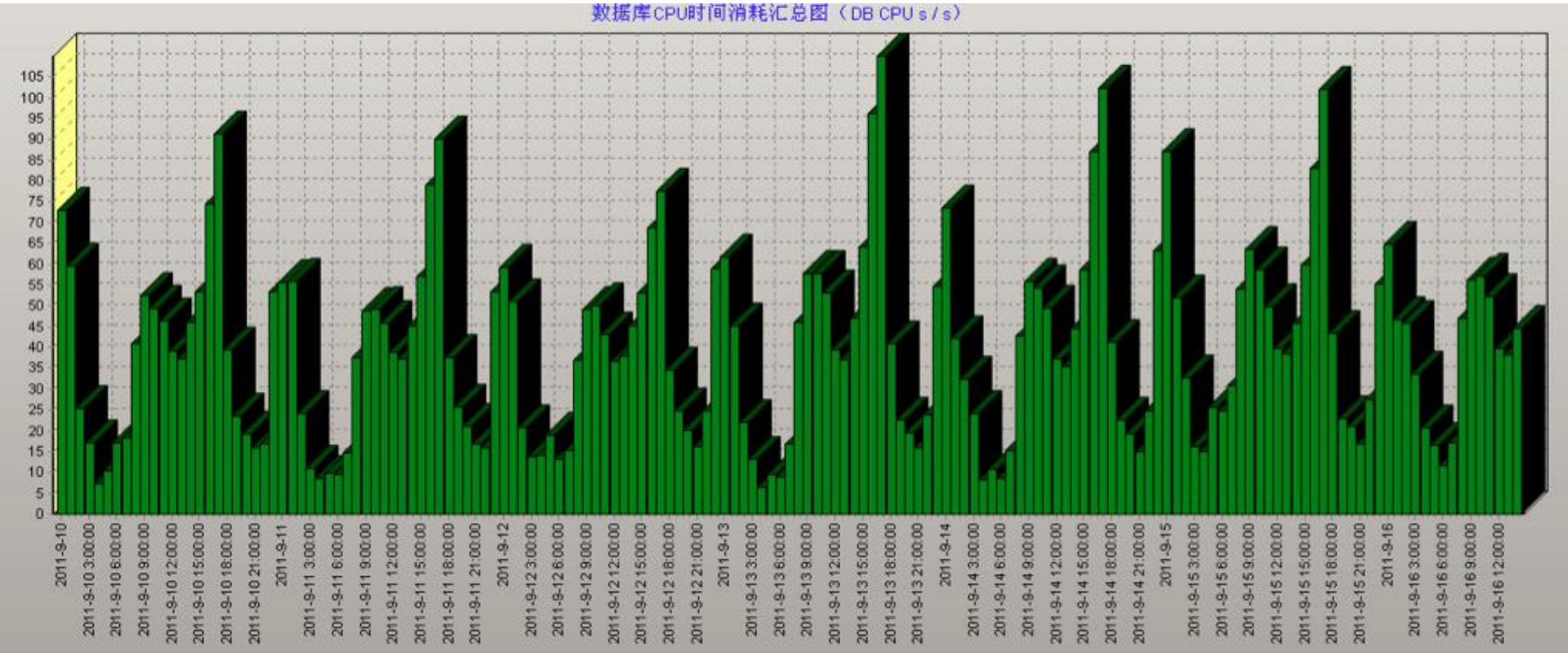
数据库的合并整合

- Exp、Expdp / Imp、ImpDP
- DB Link、CTAS、INSERT
- SQLLdr
- Transport Tablespace
- MV Refresh
- Trigger

YUNHE

金融业务整合 -数据模型性能优化设计

- 31省市数据库平台整合
  - 以数据量化
  - 以数据规划
  - 以数据预测

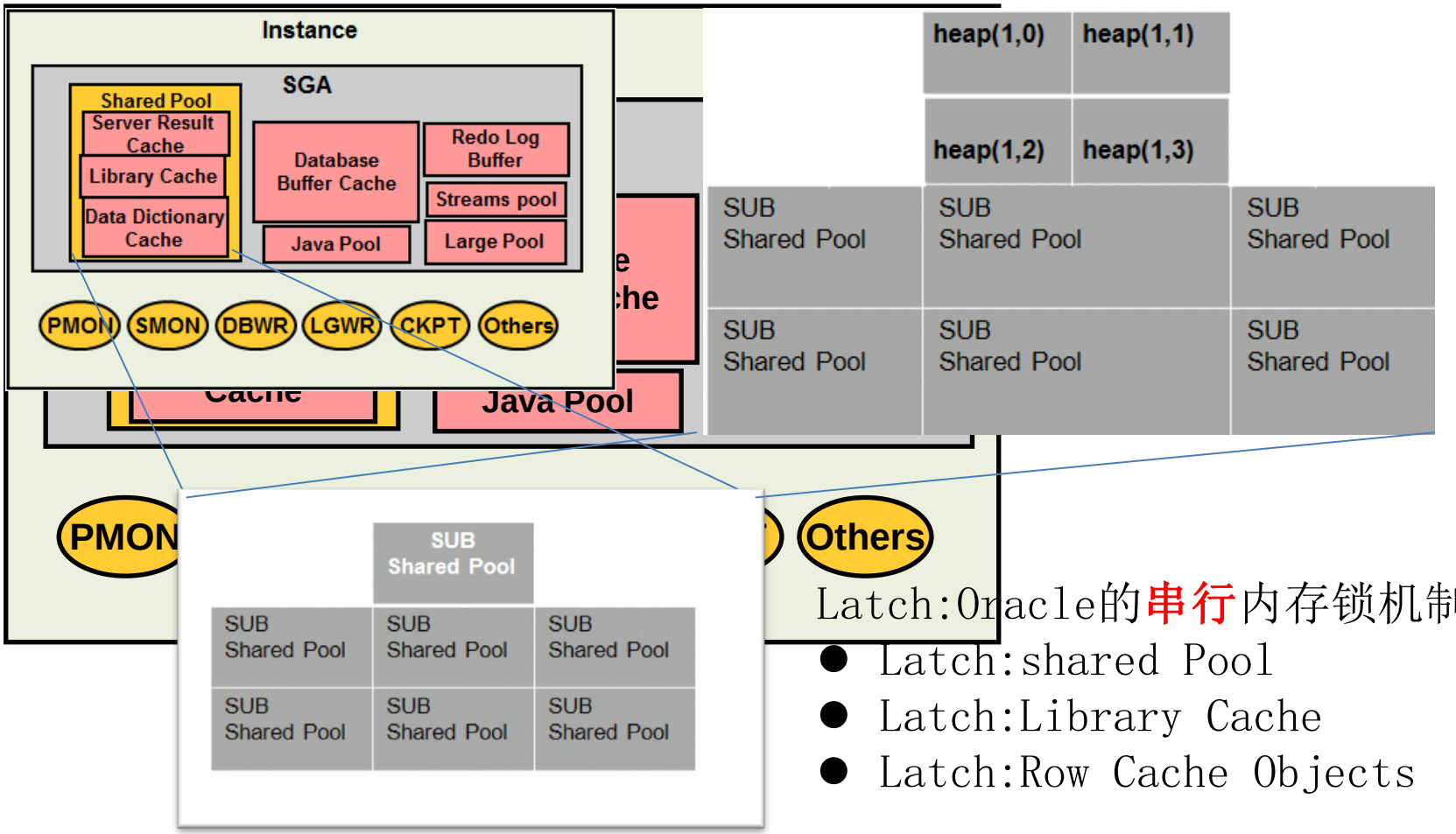


YUNHE

数据库架构的演进-分久必合

YUNHE

数据库微观的演进-拆分与并发

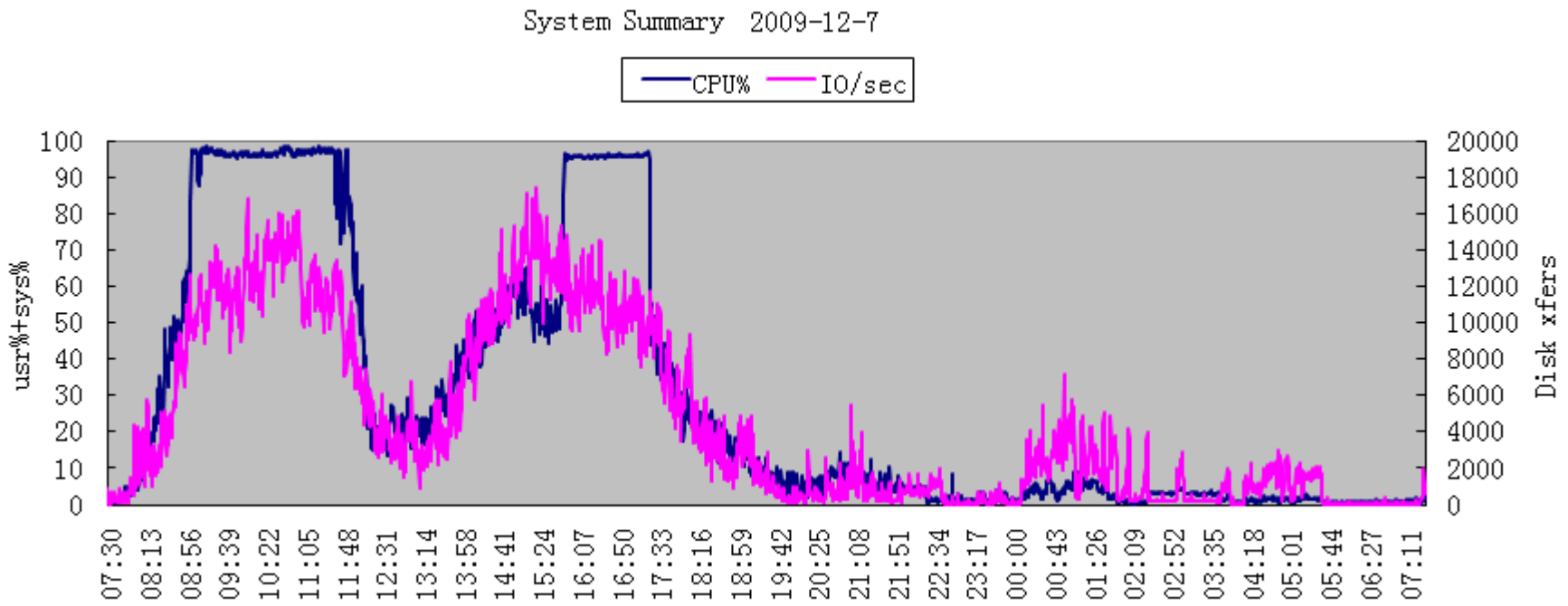


Latch:Oracle的串行内存锁机制

- Latch:shared Pool
- Latch:Library Cache
- Latch:Row Cache Objects

YUNHE

Latch竞争导致CPU 100%



	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	17170	15-Dec-09 10:30:45	1481	4.7
End Snap:	17171	15-Dec-09 10:43:58	1579	5.7
Elapsed:		13.22 (mins)		
DB Time:		7,139.78 (mins)		

YUNHE

案例分析-棘手的CPU问题

Cache Sizes

	Begin	End		
Buffer Cache:	20,480M	20,480M	Std Block Size:	8K
Shared Pool Size:	4,000M	4,000M	Log Buffer:	14,296K

Load Profile

	Per Second	Per Transaction
Redo size:	1,384,293.08	30,622.65
Logical reads:	865,863.06	19,154.20
Block changes:	7,672.89	169.74
Physical reads:	8,261.62	182.76
Physical writes:	448.08	9.91
User calls:	25,507.05	564.25
Parses:	7,753.81	171.53
Hard parses:	1,861.15	41.17
Sorts:	538.30	11.91
Logons:	0.29	0.01
Executes:	6,593.37	145.86
Transactions:	45.20	

YUNHE

案例分析-棘手的CPU问题

Cache	Get Requests	Pct Miss	Scan Reqs	Pct Miss	Mod Reqs	Final Usage
dc_awr_control	36	0.00	0		4	1
dc_files	14,220	0.00	0		0	474
dc_global_oids	840	9.40	0		0	4
dc_histogram_data	42,859,103	0.04	0		0	16,077
dc_histogram_defs	21,074,574	0.07	0		0	8,821
dc_object_grants	776	7.60	0		0	30
dc_object_ids	53,153,662	0.00	0		0	1,520
dc_objects	902,703	0.22	0		0	825
dc_profiles	4,517	0.00	0		0	1
dc_rollback_segments	29,260	0.00	0		0	829
dc_segments	5,608,352	0.06	0		22	1,689
dc_sequences	7,221	0.47	0		7,221	6
dc_table_scns	47,175	0.11	0		48	35
dc_tablespace_quotas	4	100.00	0		0	0
dc_tablespaces	1,881,653	0.00	0		0	17
dc_usernames	69,864	0.03	0		0	9
dc_users	1,695,024	0.00	0		0	53
global_database_name	6	50.00	0		0	0
outstanding_alerts	57	89.47	0		0	17

YUNHE

Oracle微观技术变革 - 合久比分

Description

It is possible for a query to have a high parse time because of high number of requests of 'latch: row cache objects' and high gets on dc_object_ids when there are materialized views present in the schema.

The reason is the query rewrite process that is trying to find matches with materialized views and it is possible that will try to match them in many different ways. For each match it needs to read the row cache several times causing high number get on dc_object_ids and requests to the latch.

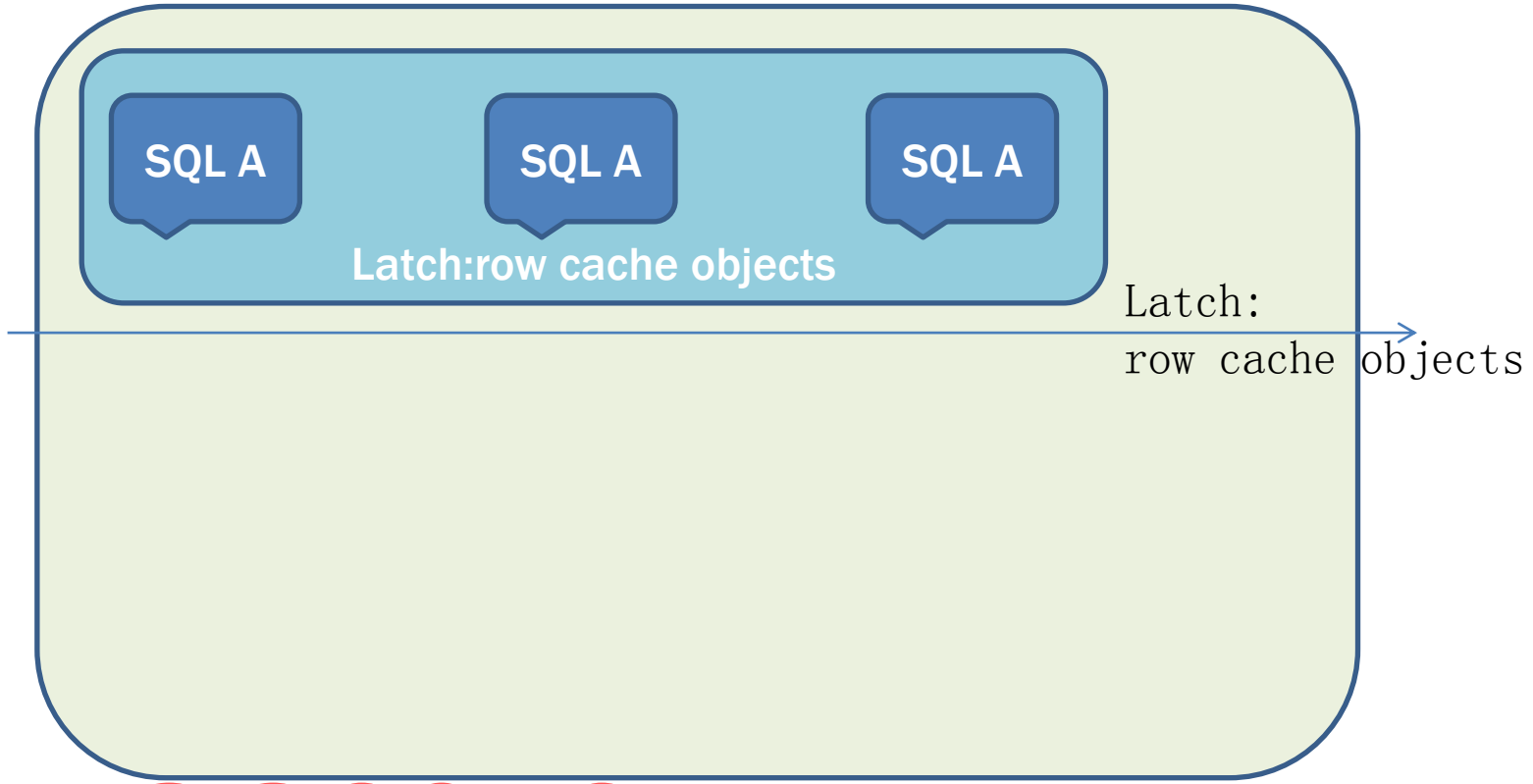
This fix is to keep a local cache to avoiding repeated lookups to dc_object_ids in the row cache.
Expect the row cache hits on dc_object_ids to be reduced but not eliminated.

In some extreme cases a dump can occur (under kkqssjsce -> .. strlen).
See bug 5572957.

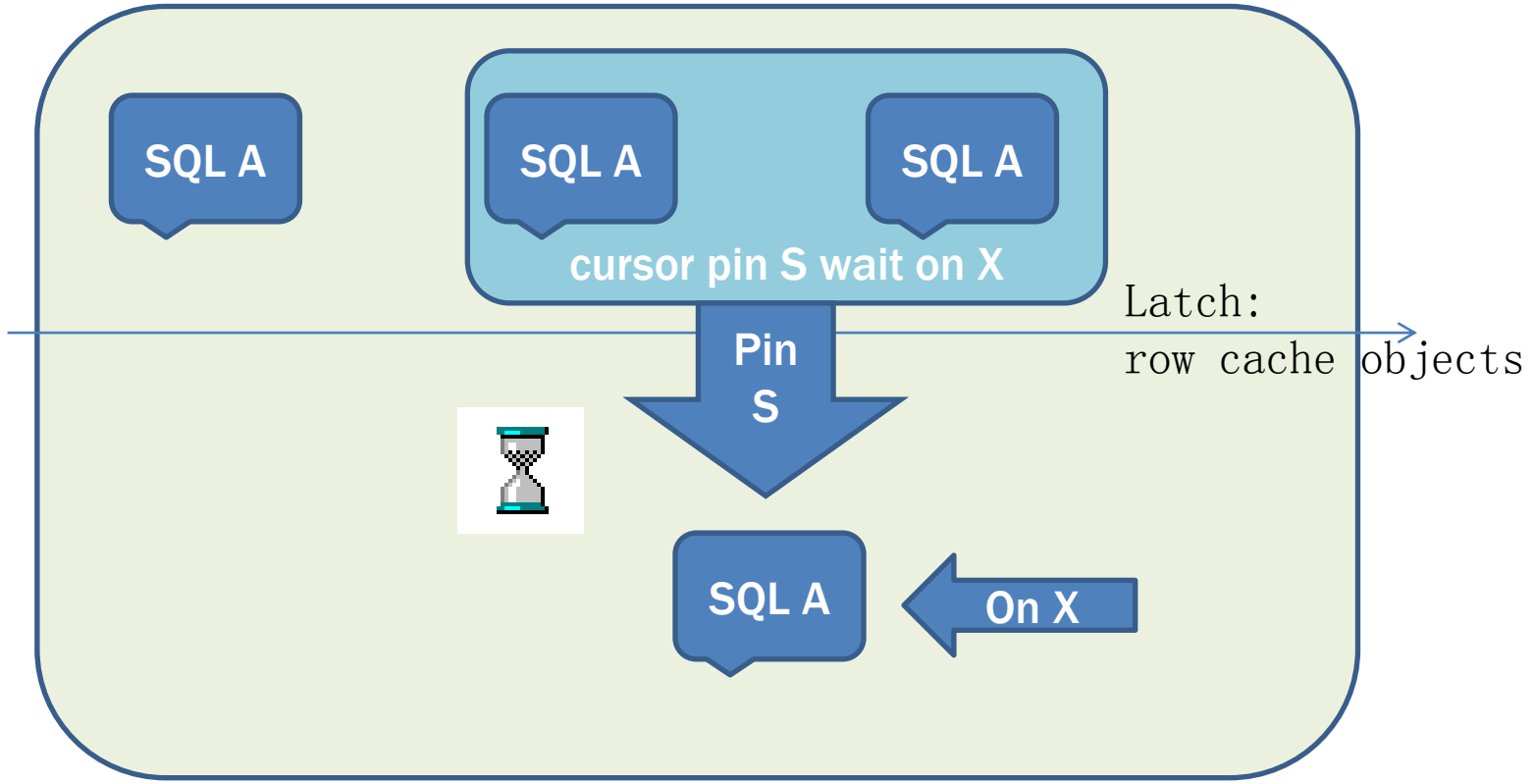
Workaround:
- session set query_rewrite_enabled=false
- Use a /*+ NOREWRITE */ hint

Please note: The above is a summary description only. Actual symptoms can vary. Matching to any symptoms here does not confirm that you are encountering this problem. Always consult with Oracle Support for advice.

Latch:row cache objects



cursor pin S wait on X



YUNHE

What mean Latch Free?

Top User Events

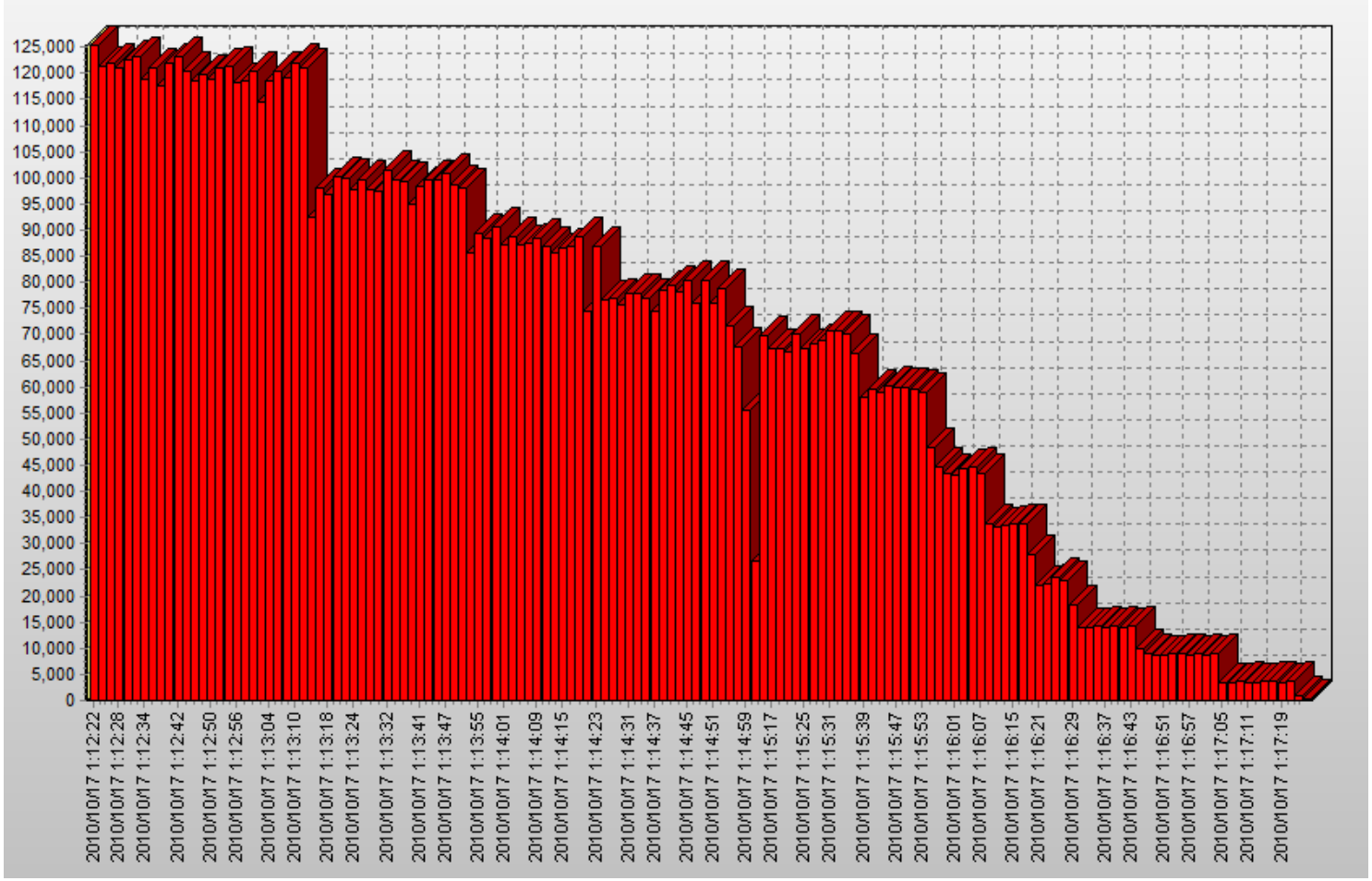
Event	Event Class	% Event	Avg Active Sessions
latch: row cache objects	Concurrency	14.89	0.08
cursor: pin S wait on X	Concurrency	12.77	0.07
db file sequential read	User I/O	4.26	0.02
rdbms ipc reply	Other	2.13	0.01

Top Event P1/P2/P3 Values

Event	% Event	P1 Value, P2 Value, P3 Value	% Activity	Parameter 1	Parameter 2	Parameter 3
latch: row cache objects	68.09	"752578468","279","0"	68.09	address	number	tries
cursor: pin S wait on X	12.77	"2935929963","720896","327680"	12.77	idn	value	where
db file sequential read	10.64	"1","70668","1"	2.13	file#	block#	blocks
		"1","85173","1"	2.13			
		"1","88762","1"	2.13			
control file sequential read	2.13	"0","1","1"	2.13	file#	block#	blocks
rdbms ipc reply	2.13	"11","1800","0"	2.13	from_process	timeout	NOT DEFINED

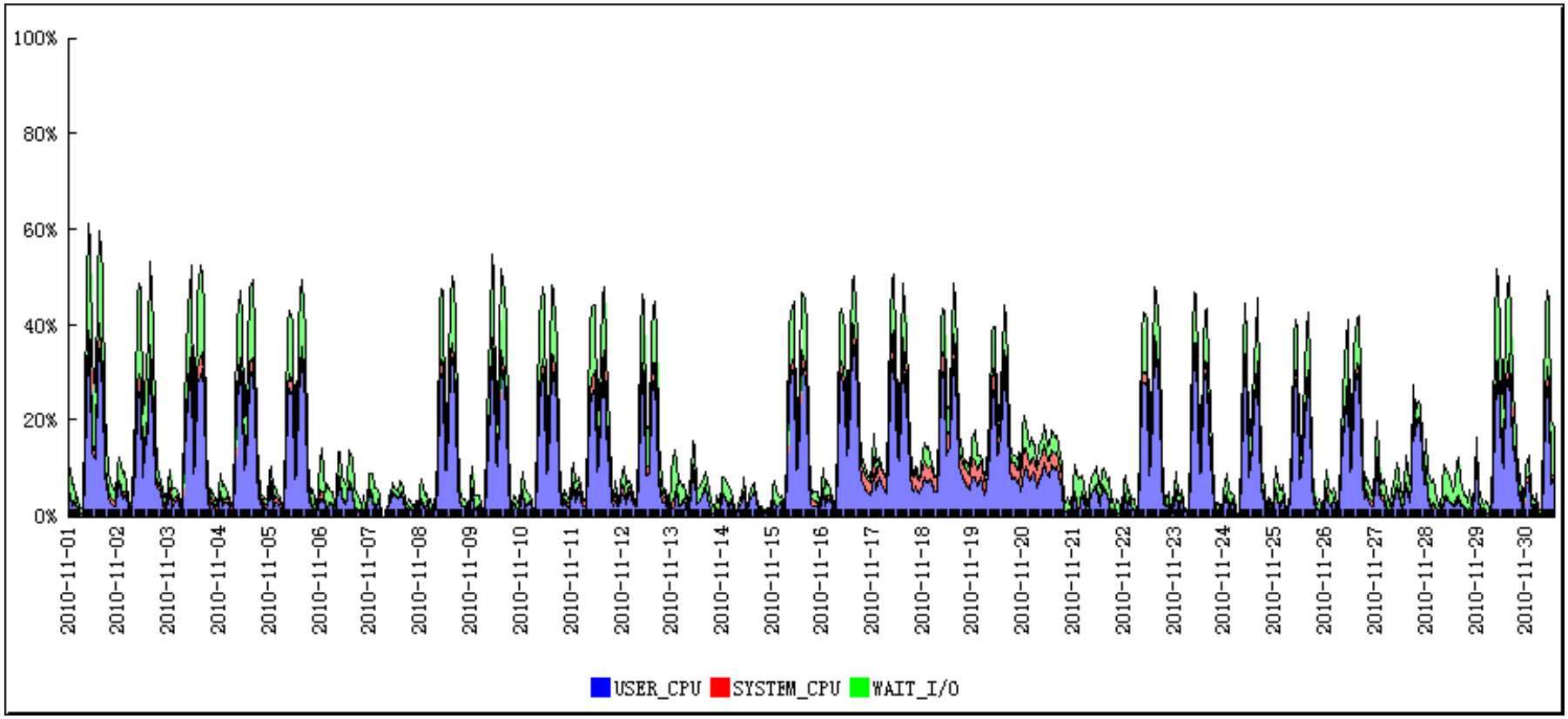
YUNHE

索引与Row Cache Gets



YUNHE

最佳实践：降低CPU负载



SACC2012

YUNHE

海量高并发 – 性能巅峰与瓶颈

Shared Pool Statistics

	Begin	End
Memory Usage %:	19.35	19.52
% SQL with executions>1:	78.50	81.79
% Memory for SQL w/exec>1:	77.83	66.90

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
log file sync	2,371,244	9,580	4	64.8	Commit
CPU time		2,645		17.9	
latch: In memory undo latch	164,632	146	1	1.0	Concurrency
log file parallel write	86,343	140	2	.9	System I/O
buffer busy waits	77,339	93	1	.6	Concurrency

YUNHE

海量高并发 – Log file Sync之痛

```
[12142]{15574} pw_postv(237, 0x9fffffffffff94e8, 0x9fffffffffff6934) ..........
[12142]{15574} write(14, "\0\0f i \0\0\0\0c0\0\001\0' bc\0"..., 96) ..........
[12142]{15574} kill(12120, SIGWINCH) ..........
[12142]{15574} read(14, "9ffffffffd; 9ba8\0\0\0\0\0\0\0\0"..., 5188) ..........
[12142]{15574} pw_postv(86, 0x9fffffffffff9030, 0x9fffffffffff66d8) ..........
[12142]{15574} write(14, "\0\0f i \0\0\0\0c0\0\001\0) \ \0"..., 96) ..........
[12142]{15574} kill(12110, SIGWINCH) ..........
[12142]{15574} read(14, "9ffffffffcw ^ c0\0\0\0\0\0\0\0\0"..., 5188) ..........
[12142]{15574} read(14, "9ffffffffcw ` p \0\0\0\0\0\0\0\0"..., 5188) ..........
( Attached to process 12826 ("ora_lgwr_tdb1") [64-bit] )
16:23:37 [12826] pw_wait(0x9fffffffffffcc20) ..........
16:23:37 [12826] (0.000121) pw_postv(79, 0x9fffffffffff8fd8, 0x9fffffffffff669c) ..........
16:23:37 [12826] (0.002648) pw_postv(10, 0x9fffffffffff8db0, 0x9fffffffffff6588) ..........
16:23:37 [12826] (0.000110) gettimeofday(0x9fffffffffffcd40, NULL) ..........
16:23:37 [12826] (0.000063) pw_post(7964685682993946721) ..........
16:23:37 [12826] (0.000105) pw_postv(84, 0x9fffffffffff9000, 0x9fffffffffff66b0) ..........
16:23:37 [12826] (0.003229) pw_postv(10, 0x9fffffffffff8db0, 0x9fffffffffff6588) ..........
16:23:37 [12826] (0.001640) pw_postv(61, 0x9fffffffffff8f48, 0x9fffffffffff6654) ..........
16:23:37 [12826] (0.002956) pw_postv(34, 0x9fffffffffff8e70, 0x9fffffffffff65e8) ..........
16:23:37 [12826] (0.000229) gettimeofday(0x9fffffffffffcd40, NULL) ..........
16:23:37 [12826] (0.000077) pw_postv(87, 0x9fffffffffff9018, 0x9fffffffffff66bc) ..........
16:23:37 [12826] (0.000773) pw_postv(7, 0x9fffffffffff8d98, 0x9fffffffffff657c) ..........
16:23:37 [12826] (0.003954) pw_postv(31, 0x9fffffffffff8e58, 0x9fffffffffff65dc) ..........
16:23:37 [12826] (0.001964) pw_postv(67, 0x9fffffffffff8f78, 0x9fffffffffff666c) ..........
16:23:37 [12826] (0.000927) pw_postv(64, 0x9fffffffffff8f60, 0x9fffffffffff6660) ..........
16:23:37 [12826] (0.003467) gettimeofday(0x9fffffffffffcd40, NULL) ..........
```

YUNHE

Log file sync - Core

LOG FILE SYNC

It's common for people to worry about the speed of writes to the log file when they see `log file sync` waits ("we're waiting for the log writer!"). If you do this, though, you will be attributing the whole of the foreground `log file sync` wait to the background `log file parallel write`.

Look back at Figure 6-3 and the session that copied commit record c4 into the log buffer and you will see a lot of activity that *isn't* about that specific record appearing between the moment the session started waiting on `log file sync` and the moment the session started running again.

If lgwr is currently writing, it will have to complete its current write and wait to get back onto the CPU, do some work with some latches, and then post a number of foreground sessions (which may push lgwr off the CPU—see the earlier note following Figure 6-3). If lgwr is not currently writing, there may still be a time lag between the foreground posting it and the moment it starts running.

Lgwr then has to check that there are commit records pending, do some work with latches, and possibly wait for some redo to be copied into the buffer, before it can finally start to write, after which it does a little more work with latches and posts the relevant foreground processes.

Once the foreground has been posted, that doesn't mean it's running, only that it's back on the run queue, and there may still be a time lag before it runs, especially if there are a lot of other processes on the system.

So if your `log file sync` times seem to be long, it's not necessarily a problem with the time it takes to write to the log files, but rather may be a problem with load on the system.

引自：Oracle Core Essential Internals for DBAs and Developers

YUNHE

海量高并发 – Log file Sync之痛

	1st per sec	2nd per sec	%Diff	1st per txn	2nd per txn	%Diff
DB time:	83.6	71.6	-14.4	0.0	0.0	0.0
CPU time:	15.0	15.4	2.6	0.0	0.0	0.0
Redo size:	29,571,917.8	25,835,486.3	-12.6	2,243.1	1,856.9	-17.2
Logical reads:	382,688.1	362,717.4	-5.2	29.0	26.1	-10.2
Block changes:	139,257.3	117,536.3	-15.6	10.6	8.4	-20.0
Physical reads:	42.7	0.6	-98.6	0.0	0.0	0.0
Physical writes:	0.6	0.2	-68.9	0.0	0.0	0.0
User calls:	13,184.3	13,914.8	5.5	1.0	1.0	0.0
Parses:	22.3	16.0	-28.4	0.0	0.0	0.0
Hard parses:	0.3	0.0	-100.0	0.0	0.0	0.0
W/A MB processed:	147,644.7	201,357.9	36.4	11.2	14.5	36.4
Logons:	0.0	0.0	-50.0	0.0	0.0	0.0
Executes:	79,133.9	83,496.4	5.5	6.0	6.0	0.0
Transactions:	13,183.3	13,913.6	5.5			

	1st Period		2nd Period	
Parameter Name	Begin Snap Value	End Snap Value	Begin Snap Value	End Snap Value
__db_cache_size		14998831104		
__java_pool_size		117440512		
__large_pool_size		67108864		
__shared_pool_size		1929379840		
__streams_pool_size		0		
_disable_logging				TRUE
audit_file_dest		/u01/oracle/admin/tdb1/adump		
background_dump_dest		/u01/oracle/admin/tdb1/bdump		

YUNHE

海量高并发 – Log file Sync之痛

Top Timed Events

- Events with a "-" did not make the Top list in this set of snapshots, but are displayed for comparison purposes

1st						2nd					
Event	Wait Class	Waits	Time(s)	Avg Time(ms)	%DB time	Event	Wait Class	Waits	Time(s)	Avg Time(ms)	%DB time
log file sync	Commit	2,371,244	9,580.34	4.04	64.83	log file sync	Commit	947,617	2,939.21	3.10	55.11
CPU time			2,645.26		17.90	CPU time			1,144.32		21.46
latch: In memory undo latch	Concurrency	164,632	146.29	0.89	0.99	latch: In memory undo latch	Concurrency	64,869	107.84	1.66	2.02
log file parallel write	System I/O	86,343	140.25	1.62	0.95	buffer busy waits	Concurrency	40,633	90.37	2.22	1.69
buffer busy waits	Concurrency	77,339	92.94	1.20	0.63	enq: TX - index contention	Concurrency	14,879	53.95	3.63	1.01
-enq: TX - index contention	Concurrency	21,562	55.49	2.57	0.38	-log file parallel write	System I/O	83,891	0.16	0.00	0.00

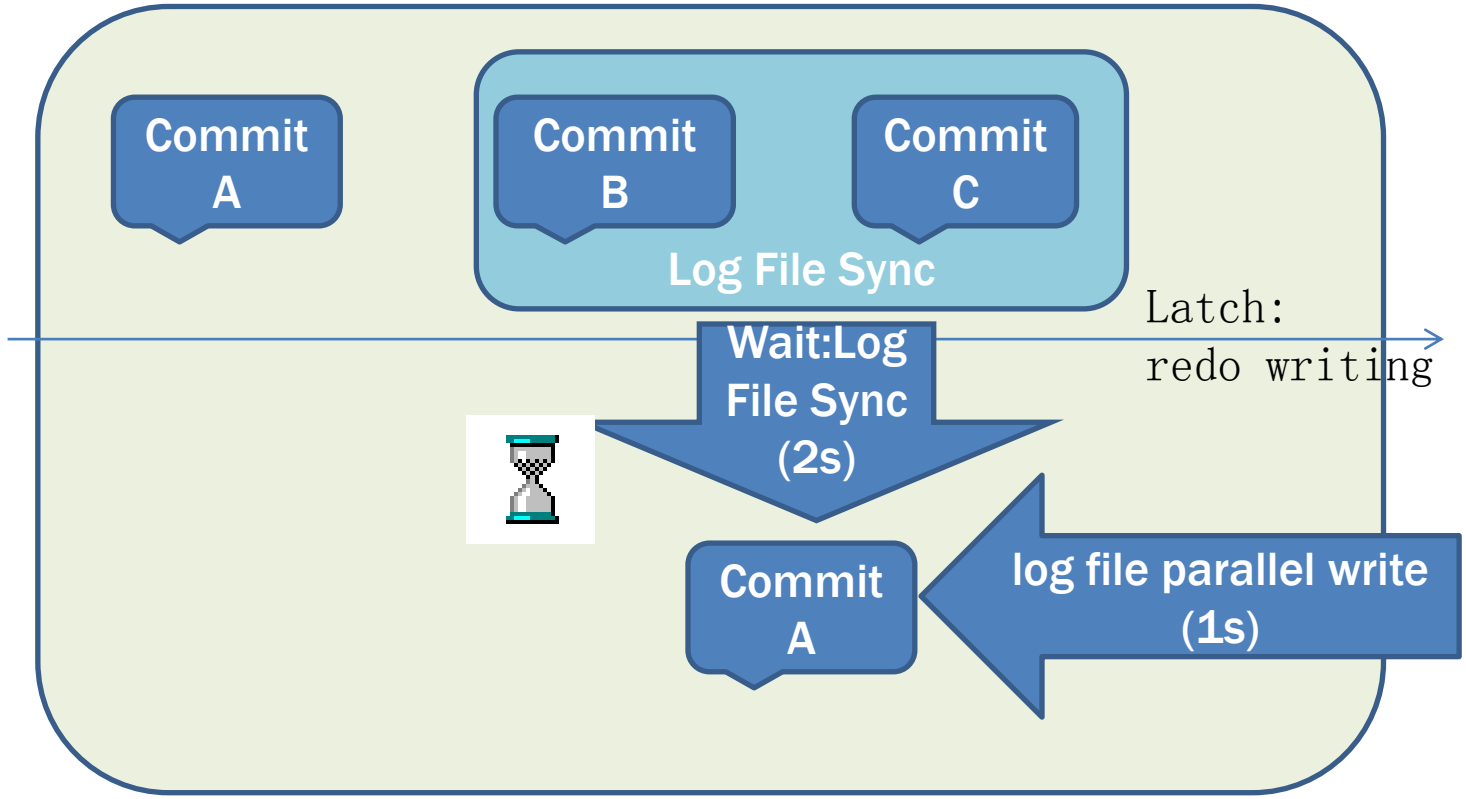
Wait Events

- Ordered by absolute value of 'Diff' column of '% of DB time' descending (idle events last)

		% of DB time			# Waits/sec (Elapsed Time)			Total Wait Time (sec)			Avg Wait Time (ms)		
Event	Wait Class	1st	2nd	Diff	1st	2nd	%Diff	1st	2nd	%Diff	1st	2nd	%Diff
log file sync	Commit	64.83	55.11	*-9.72*	13,416.11	12,721.06	*-5.18*	9,580.34	2,939.21	*-69.32*	4.04	3.10	*-23.27*
buffer busy waits	Concurrency	0.63	1.69	*1.07*	437.57	545.47	*24.66*	92.94	90.37	*-2.77*	1.20	2.22	*85.00*
latch: In memory undo latch	Concurrency	0.99	2.02	*1.03*	931.46	870.82	*-6.51*	146.29	107.84	*-26.28*	0.89	1.66	*86.52*
log file parallel write	System I/O	0.95	0.00	*-0.95*	488.51	1,126.17	*130.53*	140.25	0.16	*-99.89*	1.62	0.00	*-100.00*
enq: TX - index contention	Concurrency	0.38	1.01	*0.64*	121.99	199.74	*63.73*	55.49	53.95	*-2.78*	2.57	3.63	*41.25*
cursor: pin S	Other	0.30	0.54	*0.24*	433.45	253.02	*-41.63*	43.80	28.69	*-34.70*	0.57	1.52	*166.67*
LGWR wait for redo copy	Other	0.01	0.24	*0.23*	94.75	337.77	*256.49*	1.48	12.58	*750.00*	0.09	0.50	*455.56*
latch: library cache pin	Concurrency	0.09	0.27	*0.18*	7.21	8.81	*22.19*	12.57	14.31	*13.84*	9.86	21.81	*121.20*

Wait: Log File Sync

Wait: Log File Sync



YUNHE

总　　结

- Oracle总是在细节和架构上不断进步
- Oracle数据库的学习需要持续不断
- 高屋建瓴与细致入微成就技术人生
- 开放与分享，快乐工作，快乐生活！

SACC2012

YUNHE

Q&A