一次数据库优化全过程分析

硬件配置

Name	Value	Name	Value
Hostname	localhost	OS Type	Linux
Memory Size	32949016kB	Swap Size	18481144kB
CPU Count	8	CPU MHz	2128.052

注：该系统除了跑 oracle 外，还跑了 10 个 java 程序，每个程序大概占用 800M 内存

数据库信息

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
ORCL	1346827858	orcl	1	10.2.0.1.0	NO	localhost.localdomain

pga_aggregate_target	5368709120
processes	6000
sessions	6605
sga_max_size	21474836480
sga_target	21474836480

优化前 awr 报告信息

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	3863	06-Dec-13 08:00:57	1896	2.1
End Snap:	3864	06-Dec-13 09:01:00	1855	2.3
Elapsed:		60.06 (mins)		
DB Time:		108,269.34 (mins)		

这里可以发现 1）db time>>Elapsed，2）session 数较多

	Per Second	Per Transaction
Redo size:	211,446.31	5,999.57
Logical reads:	11,590.99	328.88
Block changes:	1,345.06	38.16
Physical reads:	5.82	0.17
Physical writes:	113.75	3.23
User calls:	737.58	20.93
Parses:	388.42	11.02
Hard parses:	0.04	0.00
Sorts:	15.15	0.43
Logons:	0.20	0.01
Executes:	392.82	11.15
Transactions:	35.24	

这里显示系统各个指标都不是非常大，证明系统不是真正忙

Top 5 等待事件

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
row cache lock	254,389	493,942	1,942	75.9	Concurrency
enq: TX - row lock contention	5,091	11,582	2,275	1.8	Application
enq: SQ - contention	11,497	11,110	966	1.7	Configuration
log file sync	426,449	9,757	23	1.5	Commit
CPU time		3,419		.5	

这里大量 row cache lock，TX，enq: SQ ，log file sync 等待

分析 enq: SQ - contention

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	638,308.40	98.04
sequence load elapsed time	617,895.85	94.91

Enqueue Type (Request Reason)	Requests	Succ Gets	Failed Gets	Waits	Wt Time (s)	Av Wt Time(ms)
TX-Transaction (row lock contention)	1,683	1,646	0	1,646	11,721	7,120.83

SQ-Sequence Cache	17,474	17,474	0	9,987	11,378	1,139.2

通过分析，证明 seq 调用确实非常频繁，需要检查 cache，并考虑增加 cache

Top sql 执行情况

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
65,613,997	463,555	141.55	76.49	1985.48	2388.82	7gtztzv329wg0		select c.name, u.name from co...
25,403,304	159,619	159.15	29.61	872.32	242310.28	07kx6r40n3fmu	JDBC Thin Client	insert into zyzhfee.np_mobile...
23,697,081	145,389	162.99	27.63	721.50	212920.96	dg54j6dm04vp0	JDBC Thin Client	insert into zyzhfee.np_client...
23,551,863	144,006	163.55	27.46	762.06	212638.86	chpbbd3k1gdm1	JDBC Thin Client	insert into zyzhfee.np_client...
2,399,608	15,101	158.90	2.80	74.49	25416.02	8wq8m7jgk49yy	JDBC Thin Client	insert into zyzhfee.np_client...

检查系统状态

```
top - 11:22:08 up 15:26,  4 users,  load average: 9.04, 7.29, 6.67
Tasks: 2113 total,  10 running, 2103 sleeping,   0 stopped,   0 zombie
Cpu(s): 14.1%us, 66.7%sy,  0.0%ni, 11.5%id,  7.4%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:  32949016k total, 32863840k used,    85176k free,     1376k buffers
Swap: 18481144k total,  6716484k used, 11764660k free,  3114184k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
  571 root      20  -5     0    0    0 R 99.9  0.0  51:21.78 [kswapd0]
  572 root      20  -5     0    0    0 R 86.0  0.0  49:58.28 [kswapd1]
16589 oracle    18   0 76144 1124  920 R 43.9  0.0   0:03.90 tar czvf dataCollector_localhost_20131206.tar dataCollector_localhost_2
16590 oracle    18   0  4044  624  252 D 34.7  0.0   0:11.80 gzip
 4994 oracle    17   0 20.1g  97m  91m R 34.0  0.3   0:02.06 oracleorcl (LOCAL=NO)
16597 oracle    19   0 98.9m  148  108 D 23.2  0.0   0:00.75 ora_m000_orcl
 3532 oracle    15   0 20.1g  99m  95m S 22.3  0.3   0:02.69 oracleorcl (LOCAL=NO)
14866 luogb     25   0  954m 302m 6180 S 21.0  0.9   1:22.28 /usr/java/jdk1.6.0_32/bin/java -Djava.util.logging.config.file=/home/lu
 5438 oracle    15   0 20.2g  56m  56m R 19.2  0.2  11:42.84 ora_lgwr_orcl
 5184 oracle    16   0 20.1g  98m  93m S 18.9  0.3   0:04.21 oracleorcl (LOCAL=NO)
```

```
[oracle@localhost ~]$ free -m
             total       used       free     shared    buffers     cached
Mem:         32176      32082         93          0          1       3046
-/+ buffers/cache:      29035       3141
Swap:        18047       7554      10493
```

```
[oracle@localhost ~]$ more /proc/meminfo
MemTotal:     32949016 kB
MemFree:         92444 kB
Buffers:          2592 kB
Cached:        2926344 kB
SwapCached:     590172 kB
Active:       17450272 kB
Inactive:       127408 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:     32949016 kB
LowFree:         92444 kB
SwapTotal:    18481144 kB
SwapFree:     10757972 kB
Dirty:           20016 kB
Writeback:           0 kB
AnonPages:    14577344 kB
Mapped:        2818160 kB
Slab:           221504 kB
PageTables:   14740844 kB
NFS_Unstable:        0 kB
Bounce:              0 kB
CommitLimit:  34955652 kB
Committed_AS: 60634280 kB
VmallocTotal: 34359738367 kB
VmallocUsed:    272032 kB
VmallocChunk: 34359466091 kB
HugePages_Total:     0
HugePages_Free:      0
HugePages_Rsvd:      0
Hugepagesize:     2048 kB
```

通过检查系统发现系统有大量的 swap 使用，内存明显不足

第一次优化分析

系统未配置大页，pagetables 占用了 14G 以上内存，内存明显不足使用了大量的 swap，而评估系统的物理读非常小，决定配置 HugePage，减小 sga，增加 seq cache 到 200

第一次优化后结果

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	3890	07-Dec-13 11:00:31	386	2.3
End Snap:	3891	07-Dec-13 12:00:33	387	2.3
Elapsed:		60.03 (mins)		
DB Time:		62.34 (mins)		

整体 db time 已经下降了很多，session 数也已经下降的很厉害，初步证明第一次优化比较成功

	Per Second	Per Transaction
Redo size:	390,991.20	3,325.74
Logical reads:	29,522.02	251.11
Block changes:	2,539.28	21.60
Physical reads:	0.09	0.00
Physical writes:	274.86	2.34
User calls:	2,774.90	23.60
Parses:	957.68	8.15
Hard parses:	0.04	0.00
Sorts:	50.30	0.43
Logons:	0.03	0.00
Executes:	977.81	8.32
Transactions:	117.57	

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		3,312		88.5	
log file sync	451,977	540	1	14.4	Commit
log file parallel write	789,519	188	0	5.0	System I/O
SQL*Net break/reset to client	1,152,660	186	0	5.0	Application
log file switch completion	265	28	106	.8	Configuration

等待事件主要变成了和 log file 相关

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
81,547,858	576,202	141.53	76.69	2174.15	2174.83	7gtztzv329wg0		select c.name, u.name from co...
30,682,401	198,544	154.54	28.85	837.10	842.80	07kx6r40n3fmu	JDBC Thin Client	insert into zyzhfee.np_mobile...
28,685,157	181,020	158.46	26.98	743.53	744.41	dg54j6dm04vp0	JDBC Thin Client	insert into zyzhfee.np_client...
28,424,227	178,618	159.13	26.73	741.08	743.07	chpbbd3k1gdm1	JDBC Thin Client	insert into zyzhfee.np_client...

3,087,083	20,021	154.19	2.90	79.60	79.70	8wq8m7jqk49yy	JDBC Thin Client	insert into zyzhfee.np_client...

Sql 里面，系统负载下降，sql 执行时间变短，因为 seq cache 增加，所以每次的逻辑读稍微减小，但是依然比较大，而且 select c.name, u.name from con$ c, cdef$ cd, user$ u where c.con# = cd.con# and cd.enabled = :1 and c.owner# = u.user# SQL 语句占据太大资源，对于该语句进行第二次优化

第二次优化分析

Top sql 中几条插入频繁的语句都有主键，而且是通过主键来控制数据一致性，而在验证数据一致性的时候，就会执行如下 sql 语句：
select c.name, u.name from con$ c, cdef$ cd, user$ u where c.con# = cd.con# and cd.enabled = :1 and c.owner# = u.user#

要想降低该语句和 insert 语句的逻辑读，就需要降低该 sql 的逻辑读，分析该 sql 执行计划

```
SQL>  select NAME,POSITION,DATATYPE,DATATYPE_STRING,VALUE_STRING,LAST_CAPTURED
  2  from V$SQL_BIND_CAPTURE where SQL_ID='&sql_id'
  3  /
Enter value for sql_id: 7gtztzv329wg0
old   2: from V$SQL_BIND_CAPTURE where SQL_ID='&sql_id'
new   2: from V$SQL_BIND_CAPTURE where SQL_ID='7gtztzv329wg0'

NAME                                                             POSITION
---------------------------------------------------------------- ----------
  DATATYPE DATATYPE_STRING
---------- ------------------------------
VALUE_STRING
--------------------------------------------------------------------------------
LAST_CAPT
---------
:1                                                                    1
         2 NUMBER
53681
10-DEC-13



SQL> set autot trace exp stat
SQL> set lines 120
SQL> select c.name, u.name from con$ c, cdef$ cd, user$ u where c.con# = cd.con# and cd.enabled = 53681 and
c.owner# = u.user#;
```

```
Execution Plan
----------------------------------------------------------

Plan hash value: 2409458995


-------------------------------------------------------------------------------
| Id  | Operation           | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |        |  3820 |   164K|     38   (6)| 00:00:01 |
|*  1 |  HASH JOIN          |        |  3820 |   164K|     38   (6)| 00:00:01 |
|   2 |   TABLE ACCESS FULL | USER$  |    64 |   896 |      3   (0)| 00:00:01 |
|*  3 |   HASH JOIN         |        |  3820 |   111K|     34   (3)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL| CDEF$  |  3820 | 34380 |     25   (0)| 00:00:01 |
|   5 |    TABLE ACCESS FULL| CON$   |  6368 |   130K|      8   (0)| 00:00:01 |
-------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------


   1 - access("C"."OWNER#"="U"."USER#")
   3 - access("C"."CON#"="CD"."CON#")
   4 - filter("CD"."ENABLED"=53681)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        150  consistent gets
          0  physical reads
          0  redo size
        598  bytes sent via SQL*Net to client
        469  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

该 sql 走全表扫描，而在 CDEF$和 CON$表中的相关 index 未被正常使用

Owner	Table Name	Tablespace	Table Type	Status	Rows	Blocks	Avg Row Len	Chain Count	Degree	Cache	Analyzed

SYS	USER$	SYSTEM	Cluster Table	VALID	64	4	83	0	1	N	2013-10-07 22:00:50
SYS	CDEF$	SYSTEM	Cluster Table	VALID	6367	108	41	0	1	N	2013-12-04 22:04:50
SYS	CON$	SYSTEM	Normal Table	VALID	6368	28	21	0	1	N	2013-12-04 22:04:46

因为这个查询本身是 insert 有主键的表的一个递归 sql 语句，出现如此效率低的情况，应该属于异常情况，查询 mos 发现

Bug 5103126 - Insert of duplicate rows with unique constraint slow (Doc ID 5103126.8)

ug 9290526 Poor plan for recursive SQL used for DML involving a UNIQUE constraint

Bug 5103126 解决方法是升级到 10.2.0.2 并打上 patch

Bug 9290526 描述版本不符合，但是也是同样类型，提供 workaround 使用 rbo

因为不想给数据库升级，而且基表不能加 hint，以前测试 sql profile 控制基表也可能不行，因此我无法使用 rbo

第二次优化后结果

尝试使用 analyze 重新收集信息

```
analyze table cdef$ compute statistics;
analyze table con$ compute statistics;
```

执行计划变为

```
SQL> select c.name, u.name from con$ c, cdef$ cd, user$ u where c.con# = cd.con# and cd.enabled = 53681 and
c.owner# = u.user#;



Execution Plan
----------------------------------------------------------
Plan hash value: 4014731293

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |     5 |   210 |    11  (10)| 00:00:01 |
|*  1 |  HASH JOIN                   |         |     5 |   210 |    11  (10)| 00:00:01 |
|   2 |   NESTED LOOPS               |         |     5 |   140 |     7   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| CDEF$   |     5 |    35 |     2   (0)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN         | I_CDEF4 |     5 |       |     1   (0)| 00:00:01 |
```

```
|   5 |    TABLE ACCESS BY INDEX ROWID| CON$     |     1 |    21 |     1   (0)| 00:00:01 |
|*  6 |     INDEX UNIQUE SCAN         | I_CON2   |     1 |       |     0   (0)| 00:00:01 |
|   7 |   TABLE ACCESS FULL           | USER$    |    64 |   896 |     3   (0)| 00:00:01 |
---------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------


   1 - access("C"."OWNER#"="U"."USER#")
   4 - access("CD"."ENABLED"=53681)
   6 - access("C"."CON#"="CD"."CON#")



Statistics
----------------------------------------------------------


          0  recursive calls
          0  db block gets
         13  consistent gets
          0  physical reads
          0  redo size
        598  bytes sent via SQL*Net to client
        469  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
rows processed
```

该 sql 的逻辑读从 150 下降到了 13，优化效果明显

再次收集 awr 信息

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	3970	10-Dec-13 18:10:19	423	2.0
End Snap:	3971	10-Dec-13 18:18:49	446	2.0
Elapsed:		8.50 (mins)		
DB Time:		3.58 (mins)		

Db time 下降到 Elapsed 一半不到

	Per Second	Per Transaction
Redo size:	343,234.65	3,069.30
Logical reads:	7,447.03	66.59
Block changes:	2,009.48	17.97

Physical reads:	0.03	0.00
Physical writes:	288.62	2.58
User calls:	2,907.96	26.00
Parses:	1,026.52	9.18
Hard parses:	0.04	0.00
Sorts:	46.61	0.42
Logons:	0.08	0.00
Executes:	1,033.69	9.24
Transactions:	111.83	

逻辑读从以前的每秒 3w 左右下降到 7 千多

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
605,127	77,057	7.85	15.94	13.54	13.54	7qtztzv329wg0		select c.name, u.name from co...
569,914	23,390	24.37	15.01	11.80	11.80	chpbbd3k1gdm1	JDBC Thin Client	insert into zyzhfee.np_client...
549,291	23,776	23.10	14.47	10.59	10.59	dq54j6dm04vp0	JDBC Thin Client	insert into zyzhfee.np_client...
548,928	26,770	20.51	14.46	15.61	15.75	07kx6r40n3fmu	JDBC Thin Client	insert into zyzhfee.np_mobile...

第一条 sql 逻辑读从整体的 70%以上下降到 16%左右，相应的 insert 语句的逻辑读从原先的 150 以上下降到 25 以下，优化效果非常明显

至此该系统绝大部分算优化完工

2013 年 12 月 10 日

惜分飞