云和恩墨
ENMOTECH

Oracle那些我们踩到心痛的坑

数据驱动 成就未来

11g — 升 & 不升？用 & 不用？

- 10g早已经明确告知了官方支持已经到期
- 11g已经被广泛应用

咬向11g螃蟹第一口的并不是你~

有担心是对的，在数据库的世界不要逞强

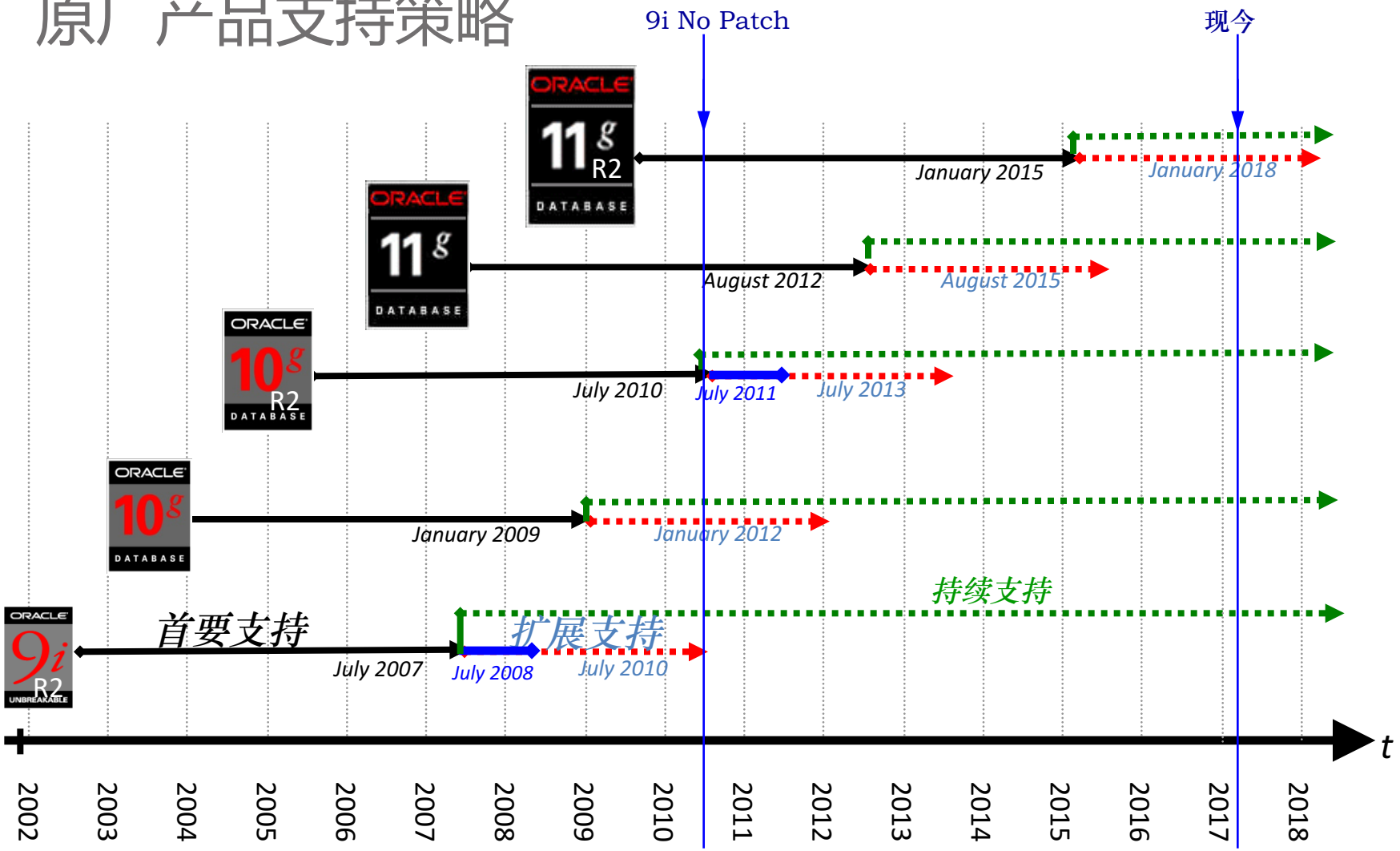
前人挖坑后人填，我们只走一马平川

问题1：要不要直接升12c？

问题2：小版本选择重要吗？

Oracle数据库产品支持策略

- 原厂产品支持策略



Oracle 11g DB新特性

- 分区增强（引用、间隔、虚拟列、更完整的组合分区）
- 活动数据卫士(ADG)
- RAC One Node
- ……

每一个大版本都带来了很多被我们所期待、所了解和不了解的特性，但并不是每一块蛋糕都像看起来那么美味，甚至有很多隐于背后的新特性是有毒的

身边的事情 — 一个10g新特性带来的烦恼

- 9i数据库，进行药品汇总信息，按药品名称统计当日发药量
- group by 带来按药品名称有序显示的统计结果
- 升级到10g 后，显示结果顺序混乱，很难在上千上万条记录中找到某一药品的统计信息

- 让一切回到原样：
  - _gby_hash_aggregation_enabled = false

一个小坑-函数的细微变化可能带来故障

- 某客户数据库升级遭遇的非规范化灾难，无有效提前验证机制
  - 数据库11.2.0.1 -> 11.2.0.3
- wmsys.wm_concat 函数
  - WMSYS用户用于Workspace Manager，其函数可能因版本而不同
  - 这种变化在11.2.0.3及10.2.0.5中体现出来。**原本WM_CONCAT函数返回值为VARCHAR2变更为CLOB。**这一变化导致了很多程序的异常

- 该函数可以实现行列转换

```
SQL> select wmsys.wm_concat(username) from dba_users;
WMSYS.WM_CONCAT(USERNAME)
--------------------------------------------------------------------------------
SYS,SYSTEM,YANGTK,TEST,OUTLN,MGMT_VIEW,FLOWS_FILES,MDSYS,ORDSYS,EXFS
```

- 在11.2.0.3中，其返回值类型变更为CLOB

```
SQL> desc wmsys.wm_concat
FUNCTION wmsys.wm_concat RETURNS CLOB
 Argument Name  Type           In/Out Default?
 ------------------------------ ------ --------
 P1       VARCHAR2             IN
```

又一个坑-ASM与内存管理变化导致的故障

多次出现ASM实例报错，该错误已经发生多次，最近的错误终于引起数据库崩溃。

下表是关于该类错误发生的时间和次数统计：

	最早发生时间	上次发生时间	最近发生时间	累计发生次数
实例1（ASM）	Fri Nov 30 22:21:32 2012	Wed Jun 26 19:42:48 2013	Tue Jul 02 16:41:43 2013	148
实例2（ASM）	Fri Nov 30 22:25:05 2012	Thu Jun 20 22:07:39 2013	Tue Jul 02 16:41:12 2013	111

故障原因：

在 Oracle 11.2.0.3中，processes的初始设置发生改变，算法改为：processes = “CPU cores * 80 + 40”，而数据库内存需要和 "processes” 有关, 因此建议增加 memory_max_target 和 memory_target 的参数设置，否则SGA不足会导致ora-04031的发生。

因此在11.2.0.3以上版本中，Oracle推荐至少设置如下内存大小，以避免ORA-04031的内存错误：

SQL> alter system set memory_max_target=4096m scope=spfile;

SQL> alter system set memory_target=1536m scope=spfile;

一个相似的故事：

11.2.0.1中，原来稳定运行的RAC+ASM架构，在做了一次“物理整合”后开始出现ora-04031.....

因为ASM实例上所支持的数据库实例增多，导致对shared pool内存空间需求增大，而原来默认的ASM SGA并不能满足需求。

还是内存惹的祸

- 某运营商，9.2.0.8库直接升级到11.2.0.4
- 当天夜里，升级顺利/业务运行顺利
- 不幸的是，第二天大量业务连接进来时，发现会话数到一定量之后无法连接

- 现象：
  - 已连接会话可正常运行
  - 系统IO量明显增加
  - free内存耗尽

- 为什么
  - 因为，11g 会话内存占用是9i的一倍
  - 数据库内存是怎么构成的：
    SGA + processes*15M + ASM SGA + ASM processes*15M(进程内存+PGA)

自动化的不都是好的-自动计划任务

- 数据库从**10.2.0.4**升级到**11.2.0.2**，升级后正常运行。
- 周末，业务运行高峰，系统突然**hang**死，大部分操作无法进行。
- 数据库中出现大量**latch free**等待和 **Resmgr:Cpu Quantum** 等待

通过现象分析，发现是resource manager plan触发的bug 10623249 - High "latch free" waits when Resource Manager is used (Doc ID 10623249.8)。

在11g，Oracle内置了resource manager plan，同时在特定的时间段（每晚22:00到早上6:00，周末全天）自动打开维护窗口的资源计划,其维护窗口可能自动会执行sql auto tunning或者segment advisor，产生大量共享池需求，而此时如果系统存在高并发压力时，将导致latch资源不足或触发相应的BUG。

那么，将资源管理计划置空，是否解决了问题？

```
alter system set resource_manager_plan='' scope=both;
```

从11g开始每个weekday window有一个预定义的Resource Plan叫做DEFAULT_MAINTENANCE_PLAN，当相关的window被打开的时候它就会被激活。

较好的解决办法是将维护窗口移动到一个较空闲的时候，以便能有足够有效的CPU资源去运行和完成这个任务。当然，也可以彻底将维护窗口选择置空的resource manager。

```
execute dbms_scheduler.set_attribute('WEEKNIGHT_WINDOW','RESOURCE_PLAN','');
execute dbms_scheduler.set_attribute('WEEKEND_WINDOW','RESOURCE_PLAN','');
```

自动化的不都是好的-自动计划任务

建议关闭11g中一些不需要的维护任务:

●这两个属于ORACLE_OCM的任务不关闭，可能会在alert日志中报错。

```
exec dbms_scheduler.disable( 'ORACLE_OCM.MGMT_CONFIG_JOB' );
exec dbms_scheduler.disable( 'ORACLE_OCM.MGMT_STATS_CONFIG_JOB' );
```

●关闭auto space advisor,避免IO消耗过多，避免该任务引起的library cache lock

```
exec DBMS_AUTO_TASK_ADMIN.DISABLE(client_name => 'auto space advisor', operation => NULL,window_name => NULL);
```

●关闭数据库的SQL自动调整Advisor，避免消耗过多的资源

```
exec DBMS_AUTO_TASK_ADMIN.DISABLE(client_name => 'sql tuning advisor', operation => NULL, window_name => NULL);
```

●调整时间窗口

```
EXECUTE
DBMS_SCHEDULER.SET_ATTRIBUTE('SATURDAY_WINDOW','repeat_interval','freq=daily;byday=SAT;byhour=22;byminute=0;bysecond=0');
```

安全增强挖下的坑

DBA很郁闷，因为每天都不得不面对一堆的客户抱怨，数据库连不上了，因为用户被锁。在以管理员身份登录数据库之后，发现用户确实被锁，于是解锁，告知可用了。

然而在一段时间之后，用户再次被锁，周而复始往复循环。DBA似乎已经习惯了听到数据库连不上了，登上去一次次解锁用户。

回头想想，曾几何时，生活变得如此？一切似乎是之前曾经做过的那次数据库升级之后。。。

何因?

出于安全性考虑，11g默认将DEFAULT的PROFILE设置登录失败尝试次数（10次）。这样在无意或恶意的连续使用错误密码连接时，数据库用户将被锁住。

一个好的思想，如果不在合适的土壤，最终得到的已经面目全非。

因此，我们不得不放弃安全性的增强，重新将登录失败尝试次数设为不限制。

一次让我们意外的宕机之坑

”我们的核心系统在晚上9点多突然有一个节点挂了，实例自动重启。虽然没有影响业务，但这种无缘无故的重启发生在核心系统上，直接威胁我们的业务运行，领导很重视，所以今天必须把原因找到，多晚都要给出结果，拜托了！“

1、重启的是整个数据库架构的2节点，这个库是核心系统，晚上也有业务；

2、重启实例的情况其实以前也发生过，只不过发生的不多；（潜台词是也许这不是个案，以前的重启意味着可能这个问题一直存在）

3、当前数据库版本为11.2.0.1。（看到每个大版本的第一个小版本，总是觉得这种系统会BUG缠身。。。虽然夸大了点，但这确实也是经验之谈。。。）

从节点2的alert log看起：

```
Fri Jun 26 21:50:26 2015
 LCK0 (ospid: 29987) waits for latch 'object queue header operation' for 77 secs.
 Errors in file /u01/app/oracle/diag/rdbms/orcl/orcl2/trace/orcl2_lmhb_29939.trc  (incident=204141):
 ORA-29771: process MMON (OSID 29967) blocks LCK0 (OSID 29987) for more than 70 seconds
Incident details in: /u01/app/oracle/diag/rdbms/orcl/orcl2/incident/incdir_204141/orcl2_lmhb_29939_i204141.trc
 MMON (ospid: 29967) is blocking LCK0 (ospid: 29987) in a wait
 LMHB (ospid: 29939) kills MMON (ospid: 29967).
 Please check LMHB trace file for more detail.
Fri Jun 26 21:51:06 2015
 Restarting dead background process MMON
 Fri Jun 26 21:51:06 2015
 MMON started with pid=213, OS id=16612
```

一次让我们意外的宕机之坑

Fri Jun 26 21:54:10 2015
LMS1 (ospid: 29929) waits for latch 'object queue header operation' for 81 secs.
LCK0 (ospid: 29987) has not called a wait for 85 secs.
Errors in file /u01/app/oracle/diag/rdbms/orcl/orcl2/trace/orcl2_lmhb_29939.trc (incident=204142):
ORA-29770: global enqueue process LMS1 (OSID 29929) is hung for more than 70 seconds
.....
Fri Jun 26 21:54:20 2015
ORA-29770: global enqueue process LCK0 (OSID 29987) is hung for more than 70 seconds
Incident details in: /u01/app/oracle/diag/rdbms/orcl/orcl2/incident/incdir_204143/orcl2_lmhb_29939_i204143.trc
ERROR: Some process(s) is not making progress.
LMHB (ospid: 29939) is terminating the instance.
Please check LMHB trace file for more details.
Please also check the CPU load, I/O load and other system properties for anomalous behavior
ERROR: Some process(s) is not making progress.
LMHB (ospid: 29939): terminating the instance due to error 29770
.....
Instance terminated by LMHB, pid = 29939
Sat Jun 27 12:52:23 2015
Starting ORACLE instance (normal)

简单描述一下过程：

1.LMHB 进程check发现mmon阻塞了LCK0进程超过70s，因此kill MMON进程，MMON进程被kill之后自动重启;

2.在Jun 26 21:54:10，LMS1进程报错无法获得latch（object queue header operation） 超过85秒;

3.在21:54:20时间点，LMHB进程强行终止了数据库实例，而终止实例的原因是LMHB进程发现LCK0进行hung住超过了70s。实际上在21:54:10时间点，LCK0进程就已经没有活动了了，而且在该时间点LMS1进程也一直在等待latch;

4.很明显，LMS1进程无法正常工作，Oracle为了保证集群数据的一致性，所以将问题节点强行驱逐重启。

一次让我们意外的宕机之坑

那么LMS1在等什么呢？LCK0为什么被Hung住了？检查一下orcl2_lmhb_29939_i204142.trc（Jun 26 21:54:10）

```
(latch info) wait_event=0 bits=a
      Location from where call was made: kcbo.h LINE:890 ID:kcbo_unlink_q_bg:
   waiting for 993cfec60 Child object queue header operation level=5 child#=117
      Location from where latch is held: kcl2.h LINE:3966 ID:kclbufs:
      Context saved from call: 0
      state=busy(shared) [value=0x4000000000000001] wlstate=free [value=0]
      waiters [orapid (seconds since: put on list, posted, alive check)]:
       14 (95, 1435326858, 4)
       21 (94, 1435326858, 7)
       waiter count=2
      gotten 73961423 times wait, failed first 4752 sleeps 1927
      gotten 33986 times nowait, failed: 4
      possible holder pid = 36 ospid=29987
```

LMS1 进程所等待的资源（object queue header operation）正被ospid=29987 持有，那么29987又是什么呢？

```
(latch info) wait_event=0 bits=20
   holding    (efd=3) 993cfec60 Child object queue header operation level=5 child#=117
      Location from where latch is held: kcl2.h LINE:3966 ID:kclbufs:
      Context saved from call: 0
      state=busy(shared) [value=0x4000000000000001] wlstate=free [value=0]
      waiters [orapid (seconds since: put on list, posted, alive check)]:
       14 (95, 1435326858, 4)
       21 (94, 1435326858, 7)
       waiter count=2
Process Group: DEFAULT, pseudo proc: 0x9b11ca008
O/S info: user: oracle, term: UNKNOWN, ospid: 29987
OSD pid info: Unix process pid: 29987, image: oracle@ebtadbsvr2 (LCK0)
```

有两个进程在等LCK0，通过分析发现：orapid=21，是dbw2进程，即数据库写进程，而orapid=14则是lms1。

注：解释一下，orapid是oracle中的进程id，而pid则是os上的进程id。

一次让我们意外的宕机之坑

- 那么我们只要知道LCK0进程为什么会hung，就知道了此次故障的原因。
- 在21：50时间点我们发现，lck0进程是被MMON进程锁阻塞了

```
(latch info) wait_event=0 bits=26
  holding    (efd=7) 993cfec60 Child object queue header operation level=5 child#=117
     Location from where latch is held: kcbo.h LINE:884 ID:kcbo_link_q:
     Context saved from call: 0
     state=busy(exclusive) [value=0x200000000000001f, holder orapid=31] wlstate=free [value=0]
     waiters [orapid (seconds since: put on list, posted, alive check)]:
      36 (82, 1435326627, 1)
      21 (81, 1435326627, 1)
      waiter count=2
  holding    (efd=7) 9b5a5d630 Child cache buffers lru chain level=2 child#=233
     Location from where latch is held: kcb2.h LINE:3601 ID:kcbzgws:
     Context saved from call: 0
     state=busy [holder orapid=31] wlstate=free [value=0]
  holding    (efd=7) 9c2a99938 Child cache buffers chains level=1 child#=27857
     Location from where latch is held: kcb2.h LINE:3214 ID:kcbgtcr: fast path (cr pin):
     Context saved from call: 12583184
     state=busy(exclusive) [value=0x200000000000001f, holder orapid=31] wlstate=free [value=0]
Process Group: DEFAULT, pseudo proc: 0x9b11ca008
O/S info: user: oracle, term: UNKNOWN, ospid: 29967
OSD pid info: Unix process pid: 29967, image: oracle@ebtadbsvr2 (MMON)
```

从trace可以看到，之前一直被等待的993cfec60 资源（Child object queue header operation ）正被 mmon进程持有。

21:50:29，LCK0在等待mmon释放资源，此时mmon出现异常，在21:51:06 mmon 进程被LMHB强制重启。重启后，由于mmon的异常，导致21:54:18该资源仍无法被lck0 进程正常持有，最终导致21:54:20LMHB进程强制重启了整个实例。

因此，**最终的罪魁祸首是mmon进程。**

一次让我们意外的宕机之坑

- MMON进程是用于AWR信息收集的，从相关trace中可以看到很多cursor包含了大量子游标：

```
Timestamp:  Current=04-05-2015 09:48:42
LibraryObject:  Address=dff3bc60 HeapMask=0000-0001-0001 Flags=EXS[0000] Flags2=[0000] PublicFlags=[0000]
 ChildTable:  size='112'
  Child:  id='0' Table=dff3cb60 Reference=dff3c5b0 Handle=2f79e908
  Child:  id='1' Table=dff3cb60 Reference=dff3c8e0 Handle=2f4e2d90
  Child:  id='2' Table=dff3cb60 Reference=df3e7400 Handle=2f8c9e90
  ......
  Child:  id='103' Table=dc86f748 Reference=dd65c6c8 Handle=29aa92f8
  Child:  id='104' Table=dc86f748 Reference=dd65c9b8 Handle=26f3a460
  Child:  id='105' Table=dc86f748 Reference=dd65ccd0 Handle=25c02dd8
NamespaceDump:
 Parent Cursor:  sql_id=76cckj4yysvua parent=0x8dff3bd48 maxchild=106 plk=y ppn=n
  Current Cursor Sharing Diagnostics Nodes:
   ......
    Child Node: 100  ID=34 reason=Rolling Invalidate Window Exceeded(2) size=0x0
     already processed:
    Child Node: 101  ID=34 reason=Rolling Invalidate Window Exceeded(2) size=0x0
     already processed:
```

- 这是为什么在20点-21点（节点2还未重启前的时段）awr报告中出现大量cursor: mutex S 的原因：

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		47,072		93.05	
cursor: mutex S	31,751,317	18,253	1	36.08	Concurrency
gc cr multi block request	359,897	1,281	4	2.53	Cluster
gc buffer busy acquire	10,465	686	66	1.36	Cluster
library cache lock	9,285	550	59	1.09	Concurrency

- 在mmon正常通过内部SQL收集系统信息时，根本不应该出现这种情况，而此时MMON进程却出现异常，这个异常看来应该是与cursor子游标过多有关了。

一次让我们意外的宕机之坑

基于上述的分析，我们最终判断，**lck0 进程出现问题的原因与cursor 子游标过多有关**。
同时，这又与在11.2.0.1版本上的child cursor总数阀值限制过高有关（实际在版本10g中就引入了该Cursor Obsolescence游标废弃特性，10g的child cursor 的总数阀值是1024，即子游标超过1024即被过期，但是这个阀值**在11g的初期版本中被移除了**。这就导致出现一个父游标下大量child cursor即high version count的发生；由此引发了一系列的版本11.2.0.3之前的cursor sharing 性能问题。这意味着版本11.2.0.1和11.2.0.2的数据库，将可能出现大量的Cursor: Mutex S 和 library cache lock等待事件这种症状，进而诱发其他故障的发生。

强烈建议11.2.0.3以下的版本应尽快将数据库版本升级到11.2.0.3以上（11.2.0.3中默认就有"_cursor_obsolete_threshold"了，而且默认值为100），或者通过_cursor_features_enabled 和106001 event来强制控制子游标过多的情况。

SQL> alter system set "_cursor_features_enabled"=34 scope=spfile;
SQL> alter system set event='106001 trace name context forever,level 1024' scope=spfile;
正常重启数据库即可。

一次诡异的SQL性能之坑

案情描述：

2015年9月末的一天，客户接连通过邮件、电话不断找我，急迫的寻求帮助。在联系上之后，告知其核心数据库突然发生了一个诡异现象，甚至导致业务系统该功能无法正常处理。

经过简单询问，发现仅仅是一条SQL导致的，而很诡异的是，这条SQL在第二次执行时，执行计划会发生了变化，导致执行效率极低，影响业务运行。

```
SELECT  A.C_PLY_NO AS C_PLY_NO, B.N_PRM AS N_PRM, A.C_BLG_DPT_CDE,.....
  FROM T_PLY_UNDRMSG A, T_PLY_BASE B, T_FIN_PLYEDR_COLDUE C
 WHERE A.C_DOC_NO = B.C_PLY_APP_NO  AND A.C_DOC_NO = C.C_PLY_APP_NO(+)
 ......
   AND A.T_APP_TM BETWEEN
       TO_DATE('2015-09-29 00:00:00', 'YYYY-MM-DD HH24:MI:SS') AND
       TO_DATE('2015-09-30 23:59:59', 'YYYY-MM-DD HH24:MI:SS')
   AND A.T_INPUT_TM BETWEEN
       TO_DATE('2015-09-29 00:00:00', 'YYYY-MM-DD HH24:MI:SS') AND
       TO_DATE('2015-09-30 23:59:59', 'YYYY-MM-DD HH24:MI:SS')
```

一次诡异的SQL性能之坑

第一次：

Optimizer goal: All rows

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			20	1	164
COUNT STOPKEY					
FILTER					
NESTED LOOPS OUTER			20	1	164
NESTED LOOPS			15	1	126
TABLE ACCESS BY INDEX ROWID	AISYS	T_PLY_UNDRMSG	13	1	77
INDEX SKIP SCAN	AISYS	IDX_PLYUNDRMSG_UN01	12	1	
TABLE ACCESS BY GLOBAL INDEX ROWID	AISYS	T_PLY_BASE	3	1	49
INDEX UNIQUE SCAN	AISYS	PK_PLY_BASE_P	2	1	
TABLE ACCESS BY GLOBAL INDEX ROWID	AISYS	T_FIN_PLYEDR_COLDUE	5	5	190
INDEX RANGE SCAN	AISYS	IDX_FINPLYEDRCOL_PLYAPPNO	3	5	

第二次：

```
Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
      0          0          0  COUNT STOPKEY (cr=0 pr=0 pw=0 time=12 us)
      0          0          0   FILTER  (cr=0 pr=0 pw=0 time=9 us)
      0          0          0    NESTED LOOPS OUTER (cr=0 pr=0 pw=0 time=9 us cost=78029 size=164328 card=1002)
      0          0          0     NESTED LOOPS  (cr=0 pr=0 pw=0 time=8 us cost=77047 size=27468 card=218)
      0          0          0      PARTITION RANGE ALL PARTITION: 1 20 (cr=0 pr=0 pw=0 time=7 us cost=76262 size=19747 card=403)
      0          0          0       PARTITION LIST ALL PARTITION: 1 7 (cr=1476837 pr=852917 pw=0 time=486948524 us cost=76262 size=19747
card=403)
      0          0          0        TABLE ACCESS FULL T_PLY_BASE PARTITION: 1 140 (cr=1786130 pr=1015408 pw=0 time=614908158 us
cost=76262 size=19747 card=403)
      0          0          0     TABLE ACCESS BY INDEX ROWID T_PLY_UNDRMSG (cr=0 pr=0 pw=0 time=0 us cost=4 size=77 card=1)
      0          0          0      INDEX RANGE SCAN PK_PLY_UNDRMSG_HIST_20131203 (cr=0 pr=0 pw=0 time=0 us cost=3 size=0
card=1)(object id 406120)
      0          0          0    TABLE ACCESS BY GLOBAL INDEX ROWID T_FIN_PLYEDR_COLDUE PARTITION: ROW LOCATION ROW LOCATION
(cr=0 pr=0 pw=0 time=0 us cost=5 size=190 card=5)
      0          0          0     INDEX RANGE SCAN IDX_FINPLYEDRCOL_PLYAPPNO (cr=0 pr=0 pw=0 time=0 us cost=3 size=0 card=5)(object id
180111)
```

一次诡异的SQL性能之坑

NO

Cardinality Feedback - 基数反馈

参考MOS Statistics (Cardinality) Feedback - Frequently Asked Questions (文档 ID 1344937.1)

用于针对统计信息陈旧、缺失直方图或虽然有直方图但Cardinality基数计算不准确的情况。

```
Note
-----
   - cardinality feedback used for this statement
```

11GR2中可以通过V$SQL_SHARED_CURSOR视图的 USE_FEEDBACK_STATS字段来表示该SQL是否使用了Cardinality Feedback特性。

禁用特性：

alter session/system set “_optimizer_use_feedback” =false;

select /*+ opt_param('_optimizer_use_feedback' 'false') cardinality(t1,1) */ count(*) from t1;

当前问题怎么解决：

select address, hash_value from v$sqlarea where sql_id = 'a6aqkm30u7p90';

ADDRESS HASH_VALUE

———— ———

C000000EB7ED3420 3248739616

exec dbms_shared_pool.purge('C000000EB7ED3420,3248739616','C');

坑哭了我们的ASM之坑

案件现场：

1.两节点RAC，11.2.0.3，ASM

2.节点2宕机，asm实例与数据库实例均挂起，ASM实例无法正常启动

告警日志内容如下：

Sat Oct 15 22:36:57 2016
NOTE: ASMB terminating
Errors in file /oracle/app/oracle/diag/rdbms/orcl/orcl2/trace/orcl2_asmb_42598.trc:
ORA-15064: communication failure with ASM instance
ORA-03113: end-of-file on communication channel
Process ID:
Session ID: 1985 Serial number: 5
Errors in file /oracle/app/oracle/diag/rdbms/orcl/orcl2/trace/orcl2_asmb_42598.trc:
ORA-15064: communication failure with ASM instance
ORA-03113: end-of-file on communication channel
Process ID:
Session ID: 1985 Serial number: 5
ASMB (ospid: 42598): terminating the instance due to error 15064

ASM日志内容如下：

Errors in file /oracle/app/11.2.0/grid/log/diag/asm/+asm/+ASM2/trace/+ASM2_lmd0_1941.trc (incident=166481):
ORA-04031: unable to allocate 3768 bytes of shared memory ("shared pool","unknown object","sga heap(1,0)","ges enqueues")
.....
ksimac callback kponrcv failed with dead inst 4 err 4031
Errors in file /oracle/app/11.2.0/grid/log/diag/asm/+asm/+ASM2/trace/+ASM2_lmon_1939.trc:
ORA-04031: unable to allocate 4192 bytes of shared memory ("shared pool","unknown object","sga heap(1,0)","qmn tasks")
LMON (ospid: 1939): terminating the instance due to error 4031

坑哭了我们的ASM之坑

检查ASM的告警日志：

Sat Oct 15 22:37:00 2016
NOTE: No asm libraries found in the system
ERROR: -5(Duplicate disk DATA:DATA_0045)
MEMORY_TARGET defaulting to 587202560.
.....
Starting ORACLE instance (normal)

而在这段信息的最开始，有一个异乎寻常的报错：
ERROR: -5(Duplicate disk DATA:DATA_0045)

根据Oracle 官方文档(Note: 1384692.1) 所述，当ASM实例碰到Duplicate disk错误时，它会选择默认参数去做启动，而不是diskgroup 的spfile。

- 根据Oracle 官方文档(note: 1370925.1),11.2.0.3 ASM内存配置应该至少1536m，才能满足多数系统的需求。那么为什么出现了这么小的ASM内存设置？
- 根据客户的安装手册显示，ASM实例通常被设为4G！
- 为何这个节点的ASM内存设置出现了问题?

在10.15 22:37:00故障时的ASM2 alert 日志中，我们发现一段特殊的错误和警告信息：

Starting up:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Real Application Clusters and Automatic Storage Management options.
ORACLE_HOME = /oracle/app/11.2.0/grid
System name: Linux
Node name: hostorcl2
Release: 2.6.32-431.el6.x86_64
Version: #1 SMP Sun Nov 10 22:19:54 EST 2013
Machine: x86_64
WARNING: using default parameter settings without any parameter file

坑哭了我们的ASM之坑

检查ASM2的信息，进一步发现：

Sat Oct 15 22:37:24 2016
NOTE: diskgroup resource ora.FRA.dg is updated
WARNING: detected duplicate paths to the same disk:
'/dev/asm-bstds8870a_data25' and
'/dev/asm-bstds8870a_data18'

其实这个信息在2016年9月20日日常维护后，重启主机ASM实例的时候就已经报了（在2016年1月1日到9月20日之间并未报过，因为在2016年并未重启过ASM实例，该错误仅在重启实例的时候会检测发现）

Tue Sep 20 03:02:58 2016
NOTE: diskgroup resource ora.FRA.dg is updated
WARNING: detected duplicate paths to the same disk:
'/dev/asm-bstds8870a_data25' and
'/dev/asm-bstds8870a_data18'

磁盘重复的两个盘对应着DATA_0045（/dev/asm-bstds8870a_data18就是导致重复的盘）

Tue Sep 20 03:02:28 2016
NOTE: No asm libraries found in the system
ERROR: -5(Duplicate disk DATA:DATA_0045)
MEMORY_TARGET defaulting to 587202560.

坑哭了我们的ASM之坑

- 何时出现了duplicate 磁盘？

查看告警日志，在2015年/dev/asm-bstds8870a_data18属于FRA_0005，甚至于在2016年6月24日的时候仍属于FRA_0005，而DATA_0045仅有1个/dev/asm-bstds8870a_data25：

Fri Jun 24 02:38:31 2016

SQL> ALTER DISKGROUP ALL MOUNT /* asm agent call crs *//* {0:0:2} */

NOTE: Diskgroups listed in ASM_DISKGROUPS are

FRA

DATA

OCRDG

NOTE: Diskgroup used for Voting files is:

OCRDG

Diskgroup with spfile:OCRDG

Diskgroup used for OCR is:OCRDG

.....

NOTE: cache opening disk 44 of grp 1: DATA_0044 path:/dev/asm-nsds8870b_data25

NOTE: cache opening disk 45 of grp 1: DATA_0045 path:/dev/asm-bstds8870a_data25

.....

NOTE: cache opening disk 5 of grp 2: FRA_0005 path:/dev/asm-bstds8870a_data18

.....

坑哭了我们的ASM之坑

在2016.7.7 DATA_0045抛出警告，被OFFLine掉，此时数据库认为DATA_0045已经异常（应该是检测到重名磁盘，也就是/dev/asm-bstds8870a_data18已经被加入DATA_0045，在属于 FRA_0005的同时又属于DATA_0045，因此系统自动offline了DATA_0045），而**数据库参数disk_repair_time被设置为100天**，即该磁盘故障100天后自动从ASM磁盘组删除：

Thu Jul 07 15:13:08 2016

WARNING: Disk 45 (DATA_0045) in group 1 mode 0x7f is now being offlined

NOTE: cache closing disk 45 of grp 1: DATA_0045

为何10月15日发生故障？

从7.7开始的100天，即10.15为该设备被删除的时间，在9.20 ASM实例重启后的提示中可以看到还剩多长时间会删除磁盘：

Tue Sep 20 03:05:42 2016

WARNING: Disk 45 (DATA_0045) in group 2 will be dropped in: (2204897) secs on ASM inst 2

Tue Sep 20 03:08:45 2016

WARNING: Disk 45 (DATA_0045) in group 2 will be dropped in: (2204714) secs on ASM inst 2

2204897秒即25.52天，从9.20算起，10.15正好25天，因此在10.15晚间发生了删除磁盘DATA_0045动作，该动作触发ASM磁盘的rebalance操作，需要较大ASM内存，而此时ASM在9.20重启后采用了默认内存参数，最终内存不足导致实例无法正常启动。

坑哭了我们的ASM之坑

Sat Oct 15 15:45:45 2016
WARNING: PST-initiated drop of 1 disk(s) in group 2(.2125370877))
SQL> alter diskgroup DATA drop disk DATA_0045 force /* ASM SERVER */
NOTE: GroupBlock outside rolling migration privileged region
NOTE: requesting all-instance membership refresh for group=2
.....
NOTE: starting rebalance of group 2/0x7eae95fd (DATA) at power 10
Starting background process ARB0
Sat Oct 15 15:45:51 2016
ARB0 started with pid=34, OS id=48980
NOTE: assigning ARB0 to group 2/0x7eae95fd (DATA) with 10 parallel I/Os
NOTE: F1X0 copy 2 relocating from 8:53 to 16:961989 for diskgroup 2 (DATA)
Sat Oct 15 22:36:38 2016
ERROR: ORA-569 thrown in ARB0 for group number 2
Errors in file /oracle/app/grid/diag/asm/+asm/+ASM2/trace/+ASM2_arb0_48980.trc:
ORA-00569: Failed to acquire global enqueue.

最后一点：

ASM实例使用默认参数启动时，alert日志存放在/oracle/app/11.2.0/grid/log/diag/asm/+asm/+ASM2/trace/；
而以spfile参数启动时，存在/oracle/app/grid/diag/asm/+asm/+ASM2/trace/

这也是为什么这个坑真的很坑的原因.....

还有很多很多……

connect角色的转变：
ALTER SESSION
CREATE SYNONYM
CREATE VIEW
CREATE DATABASE LINK

adaptive cursor sharing
无数SQL执行计划的诞生

_px_use_large_pool
默认使用shared pool作为并行执行时的消息缓冲区，诱发ORA-4031

_partition_large_extents
新建分区会给一个比较大的初始extent大小（8M）

延迟段创建导致数据操作失败

默认开启的审计撑起巨无霸的表空间

密码延迟验证导致大量无法登陆

注意密码的大小写敏感验证

怎么走上平坦的康庄大道

- 找高手朋友
- 找第三方公司&技术大拿
- 找Bethune（ https://bethune.enmotech.com/ ）

Bethune帮你发现那些坑

数据库参数分析

参数名	检查类型	检查结果
memory_target	BASIC	11,274,289,152
pga_aggregate_target	BASIC	1,690,304,512
sga_target	BASIC	8,455,716,864
db_cache_size	DIFF_FILE	INST.*: 4,664,066,048 SPFILE.1: 4,650,434,560
sessions	DIFF_FILE	INST.*: 1,560 SPFILE.1: 1,131
shared_pool_size	DIFF_FILE	INST.*: 2,969,567,232 SPFILE.1: 2,959,081,472
O7_DICTIONARY_ACCESSIBILITY	OTHER	FALSE
active_instance_count	OTHER	
aq_tm_processes	OTHER	1
archive_lag_target	OTHER	0
asm_diskgroups	OTHER	
asm_diskstring	OTHER	
asm_power_limit	OTHER	1
asm_preferred_read_failure_groups	OTHER	
audit_file_dest	OTHER	/soft/oracle/admin/hwccdb/adump

稳定性

Baymax:

警告：数据库部分参数在实例当前的值与参数文件内的设置不一致，可能导致下次重启之后参数异常，建议关注分析。不一致的参数有：db_cache_size。

提示：在11G版本中，建议关闭分区使用大的初始化区（Extent）。参考命令：alter system set "_partition_large_extents"=FALSE;

提示：在11G版本中，建议禁用自适应游标共享，将隐含参数_optimizer_extended_cursor_sharing设置为 NONE。参考命令：alter system set "_optimizer_extended_cursor_sharing"='NONE';

提示：在11G版本中，参数_optimizer_null_aware_antijoin 是在 Oracle 11g 引入的新参数，它用于解决在反连接（Anti-Join）时，关联列上存在空值（NULL）或关联列无非空约束的问题。但是该参数不稳定，存在较多的 Bug，为避免触发相关 Bug，建议关闭。参考命令：alter system set "_optimizer_null_aware_antijoin"=FALSE;

提示：在11G版本中，隐含参数_undo_autotune 负责 undo retention（即 undo 段的保持时间）的自动调整，若由 Oracle 自动负责

勇于尝试者，要有大智慧与随时应变的能力

企业守护者，耐心等待最佳实践后的一马平川

Thanks