



中亦科技
China Etek Service

Perfecting IT service and favoring clients
'success
锻造凝练IT服务 助推用户事业发展

Oracle根因分析案例分享



关于小y-技术即人生



黄远邦

就职于北京中亦安图科技股份有限公司
简称：中亦科技、中亦安图

十年以上Oracle数据库维护经验，擅长数据库架构设计、复杂故障、复杂性能问题定位和解决。
带领数十人的服务团队为客户提供数据库运维专家服务

数年来为数十家银行总行客户提供数据库专家服务
此外为航空、证券、基金、保险、运营商、政府、制造业等众多客户提供数据库维护服务

关于根因分析

- 什么是根因分析
- 根因分析的好处
- 怎么做根因分析

为什么做DBA/开发这么累 (kaixin)

- 出现问题后，没有查明根本原因，反复加班
- 没有由点带面的意识，其他系统也反复出现
- 流程上没有把控制环节往前移到开发、上线前的环节

小y的感受是：

做DBA(技术) 其实是一件非常有意思的事情
只要**基础**扎实，掌握**方法**，再结合一些丰富的**想象力**，就如何侦探破案般有意思！
让我们来经历一次**破案**之旅吧！

故障现象-夜间批量不时报ORA-12154错误

- 夜间批量, sqlldr入库不时报ORA-12154错误

```
SQL*Loader-704: Internal error: ulconnect: OCIServerAttach [0]  
ORA-12154: TNS:could not resolve the connect identifier specified
```

- 错误发生后, 重提运行成功, 原因未明
- 出现问题不立刻处理会影响到第二天白天的业务, 处理又老要夜间加班, 客户难免感觉到累

客户的疑惑:

跑了几年了, 好好的, 最近开始出现, 而且越来来越频繁, 到底怎么了? !

什么是ORA-12154错误

```
[oracle@newnew ]$ oerr ora 12154
12154, 00000, "TNS:could not resolve the connect identifier specified"
// *Cause: A connection to a database or other service was requested using
// a connect identifier, and the connect identifier specified could not
// be resolved into a connect descriptor using one of the naming methods
// configured. For example, if the type of connect identifier used was a
// net service name then the net service name could not be found in a
// naming method repository, or the repository could not be
// located or reached.
// *Action:
// - If you are using local naming (TNSNAMES.ORA file):
// - Make sure that "TNSNAMES" is listed as one of the values of the
// NAMES.DIRECTORY_PATH parameter in the Oracle Net profile
// (SQLNET.ORA)
// - Verify that a TNSNAMES.ORA file exists and is in the proper
// directory and is accessible.
// - Check that the net service name used as the connect identifier
// exists in the TNSNAMES.ORA file.
```

错误的本质

- 该错误的本质是
 - TNS别名在tnsnames.ora中找不到定义
 - 无法解析
- 测试

```
[oracle@newnew ~]$ sqlplus test/test@TNS_NAME_NO_EXISTS  
  
SQL*Plus: Release 10.2.0.1.0 - Production on Tue Oct 6 14:34:36 2015  
Copyright (c) 1982, 2005, Oracle. All rights reserved.  
  
ERROR:  
ORA-12154: TNS:could not resolve the connect identifier specified
```

- \$ORACLE_HOME/network/adminTnsnames.ora里找不到别名定义？或者没有读权限？
- 测试OK，文件的创建和修改时间没有变化。
- 那为什么失败重提以后又不报错了呢？排除！
- 设置了TNS_ADMIN环境变量，找不到tnsnames.ora文件
- 那为什么失败重提以后又不报错了呢？排除！
- 为什么跑了好几年，最近才开始出现呢？
- 经了解，做过一个存储磁盘的变更。跟这没关系啊
- 为什么不是总出现呢？

到底还有什么可能的原因呢？ 抓狂…
是不是漏掉了什么线索？ 需要回到原点

回到原点重新梳理流程

- 1、主脚本根据配置文件/home/apuser/db.cfg
配置文件中定义了username=xx, tnsnames=xx
- 2、动态生成sqlloader的脚本/home/work/sqlldr.sh
脚本内容类似 sqlldr xx/xx@TNS_NAME

3、执行sqlloader /home/work/sqlldr.sh

4、不时报ORA-12154错误，即oracle客户端无法解析TNS别名，但重提后问题解决

哪个环节可能有问题呢
要像侦探一样仔细！

1. 重点在于 `sqlldr xx/xx@TNS_NAME`

这里的TNS_NAME，有时在tnsnames.ora中找不到定义！

2. 围绕这个线索，请大家思考可能哪个环节出了问题！

有没有不起眼，被忽略和遗漏的线索呢？

做过一个存储磁盘的变更？

真相大白

业务系统A

业务系统B

/home/work从普通文件系统变为共享文件系统!

3、执行sql loader/home/work/sqlldr.sh

4、不时报ORA-12154错误，即oracle客户端无法解析TNS别名，重提后问题解决

可以回答所有问题了!

1、为什么偶尔出?

当业务系统B覆盖掉sqlldr.sh的时候，里面带的TNS_NAME指向的是业务系统B的数据库别名，自然在业务系统A的TNSNAMES.ORA中无法找到

2、为什么以前不出? 后来才出?

文件系统最近被改造为共享文件系统!

看上去一开始被忽略的线索，需要深究!

做DBA也可以像做侦探一样有意思!

下一个案例



Perfecting IT service and favoring clients 'success
锻造凝练IT服务 助推用户事业发展

最近刚刚发出的故障（数据文件头损坏）



问题-操作系统异常掉电后数据库无法启动

```
ORA-01122: database file 1 failed verification check
ORA-01110: data file 1: '/oracle/oradata/ora_system01.dbf'
ORA-01210: data file header is media corrupt
ORA-10458 signalled during: ALTER DATABASE OPEN...
```

```
Corrupt block relative dba: 0x00400001 (file 1, block 1)
Bad header found during datafile header read
```

Data in bad block:

```
type: 11 format: 2 rdba: 0x00800001
last change scn: 0x0000.00000000 seq: 0x1 flg: 0x04
spare1: 0x0 spare2: 0x0 spare3: 0x0
consistency value in tail: 0x00000b01
check value in block header: 0x8696
computed block checksum: 0x0
```

```
Rereading datafile 1 header failed with ORA-01210
```

如果是你遇到这样的错误怎么办？

#####RDBA即数据块地址

十六进制

十进制

0X00800001

8388609



#####RDBA转变成文件号、块号

```
select
```

```
dbms_utility.DATA_BLOCK_ADDRESS_FILE(8388609) file_no,  
dbms_utility.DATA_BLOCK_ADDRESS_BLOCK(8388609) block_no  
from dual;
```

file_no

block_no

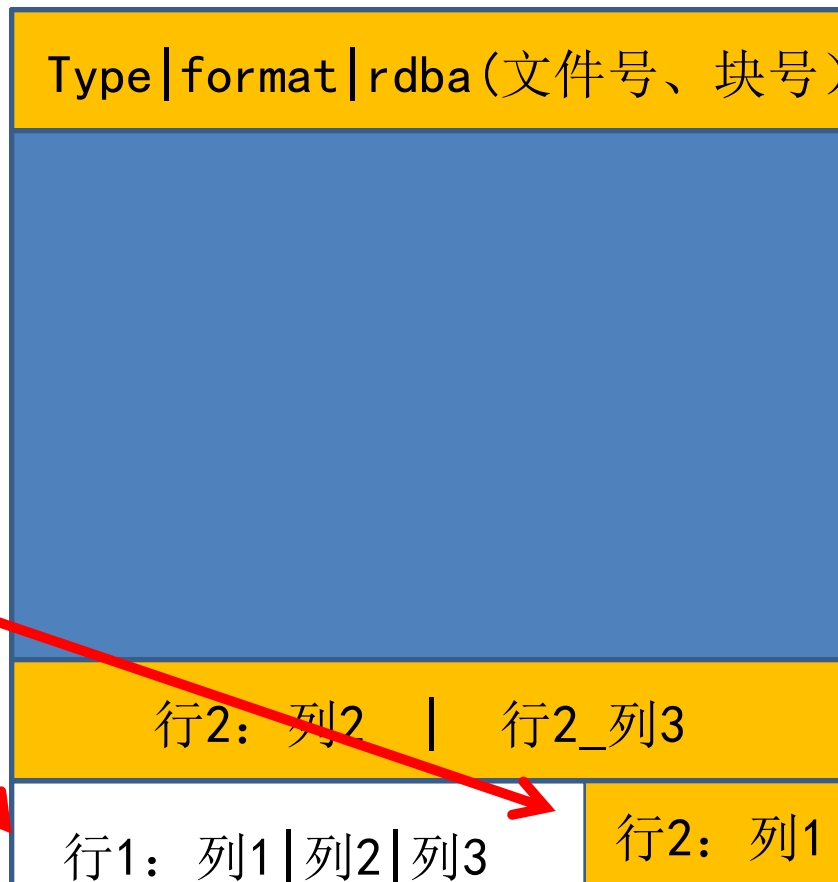
2

1

这个错误到底表示什么意思-数据块格式存储

Row directory

第1行的offset 0到20
第2行的offset 20到50
...
第n行的offset xx到yy



数据块格式存储

Oracle开始读取
1号文件1号块，
1号文件2号块

...

但是当读到1号文件1号块后，校验块的格式，发现第4到第7个字节的RDBA其实是2号文件1号块

Type format rdba (文件号、块号)	
行2: 列2 行2_列3	
行1: 列1 列2 列3	行2: 列1

遇到这样的问题要冷静！

- 1、无法找到问题原因，则问题反复发生
- 2、无法找到问题原因，盲目进行处理，使得问题变得更糟糕
- 3、无法找到问题原因，使得问题处理更发杂，业务受影响的时间更长

环境介绍和客户反馈的信息

1、基础环境：

rhel x86_64bit

oracle 11.2.0.4.6

数据文件存放在裸设备上

2、备份情况：

无备份！

3、客户初步判断：

主机重启后，操作系统把数据文件的头给重写了！！

导致1号文件文件头损坏，继而数据库无法启动！

接下来如果是你，怎么处理？

客户的选择

由于无法查到原因
数据库又无法打开
所以客户开始准备使用DUL软件对数据文件
进行抽取

（也有些客户选择设置隐含参数，强行拉库，
后面你会发现完全不可行！）

DUL bootstrap直接抽数报错，求助我们

```
DUL: Error: Wrong DBA 0X00871871 (file=2, block=465009)
DUL: Error: While processing ts# 0 file# 1 block# 465009

DUL: Error: Wrong DBA 0X00871872 (file=2, block=465010)
DUL: Error: While processing ts# 0 file# 1 block# 465010

DUL: Error: Wrong DBA 0X00871873 (file=2, block=465011)
DUL: Error: While processing ts# 0 file# 1 block# 465011

DUL: Error: Wrong DBA 0X00871874 (file=2, block=465012)
DUL: Error: While processing ts# 0 file# 1 block# 465012

DUL: Error: Wrong DBA 0X00871875 (file=2, block=465013)
DUL: Error: While processing ts# 0 file# 1 block# 465013

DUL: Error: Wrong DBA 0X00871876 (file=2, block=465014)
DUL: Error: While processing ts# 0 file# 1 block# 465014

DUL: Error: Wrong DBA 0X00871877 (file=2, block=465015)
DUL: Error: While processing ts# 0 file# 1 block# 465015

DUL: Error: Wrong DBA 0X00871878 (file=2, block=465016)
DUL: Error: While processing ts# 0 file# 1 block# 465016

DUL: Error: Wrong DBA 0X00871879 (file=2, block=465017)
DUL: Error: While processing ts# 0 file# 1 block# 465017

DUL: Error: Wrong DBA 0X0087187A (file=2, block=465018)
DUL: Error: While processing ts# 0 file# 1 block# 465018

DUL: Error: Wrong DBA 0X0087187B (file=2, block=465019)
DUL: Error: While processing ts# 0 file# 1 block# 465019

DUL: Error: Wrong DBA 0X0087187C (file=2, block=465020)
DUL: Error: While processing ts# 0 file# 1 block# 465020

DUL: Error: Wrong DBA 0X0087187D (file=2, block=465021)
DUL: Error: While processing ts# 0 file# 1 block# 465021

DUL: Error: Wrong DBA 0X0087187E (file=2, block=465022)
DUL: Error: While processing ts# 0 file# 1 block# 465022
```

至此原因基本定位



Perfecting IT service and favoring clients 'success
锻造凝练IT服务 助推用户事业发展

分析过程-梳理数据文件和裸设备的link关系

```
[oracle@oradb2 yang]$ ls -l /oracle/oradata/ora_system01.dbf  
lrwxrwxrwx. 1 oracle dba 13 Aug 22 2015 /oracle/oradata/ora_system01.dbf -> /dev/raw/raw1
```


分析过程—梳理裸设备和块设备的映射关系

```
[oracle@oradb2 yang]$ raw -qa  
/dev/raw/raw1: bound to major 253, minor 5  
/dev/raw/raw2: bound to major 253, minor 6  
/dev/raw/raw3: bound to major 253, minor 7  
/dev/raw/raw4: bound to major 253, minor 8  
/dev/raw/raw5: bound to major 253, minor 9
```

```
[oracle@oradb2 yang]$ raw -qa  
/dev/raw/raw1: bound to major 253, minor 5  
/dev/raw/raw2: bound to major 253, minor 6  
/dev/raw/raw3: bound to major 253, minor 7  
/dev/raw/raw4: bound to major 253, minor 8  
/dev/raw/raw5: bound to major 253, minor 9
```

开始发现异常

```
--- Logical volume ---
LV Path                /dev/vg_oradb/ora_sysaux01.dbf
LV Name                ora_sysaux01.dbf
VG Name                vg_oradb
LV UUID                5sWJ8J-DYf0-w0mH-bdYx-hfTe-VU7G-eS4eQc
LV Write Access        read/write
LV Status              available
# open                 0
LV Size                16.00 GiB
Current LE             4096
Segments               1
Allocation             inherit
Block device           253:5
```

异常数据（续）

```
[root@oradb2 rules.d]# lvdisplay
--- Logical volume ---
LV Path                /dev/vg_oradb/ora_system01.dbf
LV Name                 ora_system01.dbf
VG Name                 vg_oradb
LV UUID                 062V7h-X67j-VMxh-IjB3-kEIo-8Ar2-qGoHCQ
LV Write Access         read/write
LV Status                available
# open                  0
LV Size                 16.00 GiB
Current LE              4096
Segments                1
Allocation              inherit
Read ahead sectors      auto
Block device            253:4
```

为什么重启后问题出现

```
[root@oradb2 rules.d]# cat 60-raw.rules
ACTION=="add", KERNEL=="/dev/mapper/vg_oradb-ora_system01.dbf" ,RUN+="/bin/raw /dev/raw/raw1 %N"
ACTION=="add", ENV{MAJOR}=="253", ENV{MINOR}=="5", RUN+="/bin/raw /dev/raw/raw1 %M %m"
ACTION=="add", KERNEL=="/dev/mapper/vg_oradb-ora_sysaux01.dbf" ,RUN+="/bin/raw /dev/raw/raw2 %N"
ACTION=="add", ENV{MAJOR}=="253", ENV{MINOR}=="6", RUN+="/bin/raw /dev/raw/raw2 %M %m"
ACTION=="add", KERNEL=="/dev/mapper/vg_oradb-ora_sysaux02.dbf" ,RUN+="/bin/raw /dev/raw/raw3 %N"
ACTION=="add", ENV{MAJOR}=="253", ENV{MINOR}=="7", RUN+="/bin/raw /dev/raw/raw3 %M %m"
ACTION=="add", KERNEL=="/dev/mapper/vg_oradb-ora_ctrlfile01.dbf" ,RUN+="/bin/raw /dev/raw/raw4 %N"
ACTION=="add", ENV{MAJOR}=="253", ENV{MINOR}=="8", RUN+="/bin/raw /dev/raw/raw4 %M %m"
ACTION=="add", KERNEL=="/dev/mapper/vg_oradb-ora_ctrlfile02.dbf" ,RUN+="/bin/raw /dev/raw/raw5 %N"
ACTION=="add", ENV{MAJOR}=="253", ENV{MINOR}=="9", RUN+="/bin/raw /dev/raw/raw5 %M %m"
```

总结和预防



- 1、通过根因分析，避免了长时间停机和数据可能不一致的问题
- 2、如何预防？ dataGuard /RAC ？

客户的邮件

今天早上，同事反映一个问题也有点奇怪，他说批处理的应用一直挂在那里不往下走。我看了一下Exadata的数据库，我觉得是应用没有再发起下一步的处理，所以导致了这个现象，但同事认为是数据库没有返回信息给应用，导致应用那边一直等待。

想问问你怎么看？

客户抓下来的证据

```
select serial#, sql_id, event  
  from gv$session  
 where sid = 7350;
```

SERIAL#	SQL_ID	EVENT
33459	0mss26rs43c7p	SQL*Net message from client

.....

SERIAL#	SQL_ID	EVENT
33459	0mss26rs43c7p	SQL*Net message from client

如果是你，你接下来怎么查？

你亲眼看见的不一定是真实的

```
select to_char(SAMPLE TIME, 'yyyymmdd hh24:mi:ss') as sample_time,  
       event, SESSION STATE sql_id  
from dba_hist_active_sess_history  
where SAMPLE_TIME > sysdate - 0.5  
      and SESSION_ID = 7350  
      and SESSION_SERIAL# = 33459  
order by SAMPLE_TIME
```

SAMPLE_TIME	EVENT	SESSION	SQL_ID
20151221 05:21:34	ON CPU	f5grdjhv2996p	
20151221 05:21:44	ON CPU	0mss26rs43c7p	
20151221 05:21:54	ON CPU	0mss26rs43c7p	
.....			
20151221 11:40:11	ON CPU	0mss26rs43c7p	
20151221 11:40:21	ON CPU	0mss26rs43c7p	
20151221 11:40:31	ON CPU	0mss26rs43c7p	
20151221 11:40:41	ON CPU	0mss26rs43c7p	
20151221 11:40:51	ON CPU	0mss26rs43c7p	

执行计划

Plan hash value: 3759901922

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	INSERT STATEMENT					33G(100)	
1	LOAD TABLE CONVENTIONAL						
2	FILTER						
3	HASH GROUP BY		370T	607P		33G(100)	999:59:59
4	HASH JOIN		370T	607P	318M	1777M (83)	999:59:59
5	TABLE ACCESS STORAGE FULL	TB XX EXADATA	2932K	285M		30765 (1)	00:04:07
6	TABLE ACCESS STORAGE FULL	TB XX EXADATA	2650M	4307G		61191 (50)	00:08:10

SQL语句

```
INSERT INTO TB_XX_EXADATA_HIS
(TEMPKEY,
DATEDT,
HALFRESULT,
FCETKEY,
FCETTYP ECODE,
FCETNAME,
ORGANKEY,
ALERTDESC)
SELECT t2.TEMPKEY,
      t2.DATEDT,
      t2.HALFRESULT,
      '1102-020603',
      t2.FCETTYPECODE,
      t2.FCET NAME,
      t2.ORGANKEY,
      t2.ALERTDESC
FROM TB_XX_EXADATA t2,
     (SELECT distinct t.tempkey
      FROM TB_XX_EXADATA t
      GROUP BY t.tempkey
      HAVING count(t.tempkey) >= 10000) t1
WHERE t2.tempkey = t1.tempkey
```

思考一下什么原因

插播一个案例，你就会明白了😊

案例-突然变慢且再也快不回来的SQL

问题描述

- 故障现象
 - 应用有一条SQL语句，平时跑10分钟，10月20日起跑10个小时以上。现象可重现
 - 数据量无明显变化
 - 收集统计信息，重启数据库均无法恢复到原来的执行时间
 - 运维DBA和开发均介入，原因未明
- 如果case转到了你手里（你就是运维DBA或开发），你该怎么查
 - 怎么解决
 - 为什么以前不出，而是某一天后开始，以后还会不会再出（领导关心）

完整的SQL语句

```
SELECT /*+ FULL(SMALL_TABLE) USE_HASH(SMALL_TABLE, BIG_TABLE) */  
      BIG_TABLE.COL,  
      .....  
      SUM(SMALL_TABLE.COL2) SUM1,  
      SUM(SMALL_TABLE.COL3) SUM2  
FROM BIG_TABLE   BIG_TABLE, -----3500M , 800万  
      SMALL_TABLE SMALL_TABLE ---80M , 160万  
WHERE SMALL_TABLE.ID = BIG_TABLE.ID -----关联条件  
AND .....  
GROUP BY BIG_TABLE.COL2;
```

...
可以看到：两张表关联，然后group by
SQL语句用了hint(告诉数据库，走什么样的执行计划)

SQL执行的相关统计

Stat Name	Statement	Per Execution	% Snap
Elapsed Time (ms)	3.9616E+07	39,615,580.6	42.3
CPU Time (ms)	3.8719E+07	38,719,288.3	61.3
Executions	1	N/A	N/A
Buffer Gets	452,276	452,276.0	0.0
Disk Reads	451,421	451,421.0	0.5
Parse Calls	1	1.0	0.0
Rows	514	514.0	N/A
User I/O Wait Time (ms)	32,751	N/A	N/A
Cluster Wait Time (ms)	353	N/A	N/A
Application Wait Time (ms)	1	N/A	N/A
Concurrency Wait Time (ms)	0	N/A	N/A
Invalidations	0	N/A	N/A
Version Count	45	N/A	N/A
Sharable Mem(KB)	2,185	N/A	N/A

每次执行时间39,615秒

每次执行逻辑读45,276个block(块)

每次执行物理读451421个block(块)

时间基本都消耗在CPU上,而在IO/集群/应用(锁)/并发环节基本没有发生什么等待事件,消耗时间很小

执行计划

Execution Plan

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT					49636 (100)	
1	HASH GROUP BY		328	19680		49636 (3)	00:25:36
2	VIEW	VW_0	1725K	98M		49567 (3)	00:25:34
3	HASH GROUP BY		1725K	83M	126M	49567 (3)	00:25:34
4	HASH JOIN		1725K	83M	56M	31091 (5)	00:16:02
5	TABLE ACCESS FULL	SMALL_TABLE	1698K	37M		633 (6)	00:00:20
6	PARTITION LIST SINGLE		4216K	112M		27093 (5)	00:13:59
7	TABLE ACCESS FULL	BIG_TABLE	4216K	112M		27093 (5)	00:13:59

可以看到，执行计划（oracle内部的算法）确实如hint一样

表连接方式走的是hash join

单表访问路径都是全表扫描(table access full)

表连接顺序是小表做驱动表(hash内存表)

其他线索1

```
sys@OBIE> select  sofar,totalwork from v$session_longops  
where sofar != totalwork and sid=1614 ;
```

SOFAR	TOTALWORK
362690	442460

```
sys@OBIE> exec dbms_lock.sleep(5);  
PL/SQL procedure successfully completed.
```

```
23:05:11 sys@OBIE> select  sofar,totalwork from v$session_longops  
where sofar != totalwork and sid=1614 ;
```

SOFAR	TOTALWORK
362691	442460

- v\$session_longops中表SMALL_TABLE已经扫描完成 100%
- 但另外一张表BIG_TABLE全表扫描的进度进本停留在82%，查看，发现每5秒才1个块

其他线索2-历史执行统计

时间	执行次数	逻辑读	物理读	执行时间	CPU时间	IO时间	CLUSTER时间	AP时间	并发时间	单次执行时间 (秒)
2015102000	1	444573	443052	3329	3170	56	0	0	0	3329
2015102001	0	1932	139	3585	3511	0	0	0	0	1
2015102002	0	1559	96	3580	3513	0	0	0	0	1
2015102003	0	1438	0	3654	3548	0	0	0	0	1
2015102004	0	1188	57	3575	3473	0	0	0	0	1
2015102005	0	1314	17	3597	3531	0	0	0	0	1
2015102006	0	1701	63	3732	3559	0	0	0	0	1
2015102007	0	1443	10	3445	3324	0	0	0	0	1
2015102008	0	1347	5	3608	3540	0	0	0	0	1
2015102009	0	1252	0	3646	3497	0	0	0	0	1
2015102010	0	258	32	872	830	0	0	0	0	1
2015102012	1	371540	370854	663	615	31	0	0	0	663
2015102013	0	1194	1184	3580	3486	0	0	0	0	1
2015102014	0	1396	1408	3609	3528	0	0	0	0	1
2015102015	0	1104	1120	3566	3485	0	0	0	0	1
2015102016	0	1315	1312	3620	3517	0	0	0	0	1
2015102017	0	1600	1600	3600	3528	0	0	0	0	1
2015102018	0	1183	1184	3598	3533	0	0	0	0	1
2015102019	0	2963	2968	3592	3507	0	0	0	0	1
2015102020	0	1894	1888	3605	3527	0	0	0	0	1
2015102021	0	871	864	3618	3535	0	0	0	0	1

- 每个小时才处理1000-3000的逻辑读
- 一开始快，后来慢
- 时间都在CPU上

其他线索3-CallStack

```
22:28:02 sys@OBIE> oradebug short_stack
```

```
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-expepr()+100<-evaor()+88<-expepr()+100<-evacssr()+168<-qerghRow
```

```
22:28:08 sys@OBIE> oradebug short_stack
```

```
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-qerstRowP()+520<-qerhjWalkHashBucket()+596<-qerhjInnerProbeHash
```

```
22:28:45 sys@OBIE> oradebug short_stack
```

```
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-qerghAggregateRecords()+528<-qeshLoadRowForGBY()+3020<-qerghRow
```

思考原因

综合所有现象，得到线索，定位原因

CallStack

```
22:28:02 sys@OBIE> oradebug short_stack
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-expepr()+100<-evaor()+88<-expepr()+100<-evacssr()+168<-qerghRow
22:28:08 sys@OBIE> oradebug short_stack
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-qerstRowP()+520<-qerhjWalkHashBucket()+596<-qerhjInnerProbeHash
22:28:45 sys@OBIE> oradebug short_stack
ksedsts()+360<-ksdxfstk()+44<-ksdxcb()+3384<-sspuser()+116<-47dc<-qerghAggregateRecords()+528<-qeshLoadRowForGBY()+3020<-qerghRow
```

Hash Join原理

```
SELECT * FROM A,B  
WHERE A.ID=B.ID
```

- 1) SCAN A
- 2) HASH(A.ID),打散到各个桶 (BUCKET) 中，呆在pga hash area中等待别人来匹配
- 3) SCAN B
- 4) HASH(B.ID)
- 5) 到相应的Bucket中，比较表关联字段，返回或丢弃

HASH的目的是为了打算数据到各个桶中

那么HASH JOIN有什么缺点呢？

我们是否命中了该缺点？！

Hash 内存表（驱动表）表关联字段分布不均

```
select *  
  from (select ID, count(*)  
        from SMALL_TABLE  
        group by ID  
        order by 2 desc);
```

ID	COUNT(*)
0	174882
9371713	8697
9348322	1506
2598178	275
10363405	168
9971658	151
20335682	144
655287	140

进一步验证Hash Join的缺点

```
SELECT /*+ FULL(SMALL_TABLE) USE_HASH(SMALL_TABLE, BIG_TABLE) */
      BIG_TABLE.COL,
      .....
      SUM(SMALL_TABLE.COL2) SUM1,
      SUM(SMALL_TABLE.COL3) SUM2
FROM BIG_TABLE   BIG_TABLE, -----3500M , 800万
      SMALL_TABLE SMALL_TABLE ---80M , 160万
WHERE SMALL_TABLE.ID = BIG_TABLE.ID -----关联条件
AND .....
      AND SMALL_TABLE.ID != 0
GROUP BY BIG_TABLE.COL2;
      ...
```


- ◆掌握原理是必须的
- ◆什么样的架构/存储结构决定了他可以做什么样的事情，不可以做什么样的事情
- ◆但你思考过他的缺点是什么么？以前没有？
建议尝试，让你有更多收获

回到上一个案例再来看原因



SQL语句

```
INSERT INTO TB_XX_EXADATA_HIS
(TEMPKEY,
DATEDT,
HALFRESULT,
FCETKEY,
FCETTYP ECODE,
FCETNAME,
ORGANKEY,
ALERTDESC)
SELECT t2.TEMPKEY,
      t2.DATEDT,
      t2.HALFRESULT,
      '1102-020603',
      t2.FCETTYPECODE,
      t2.FCET NAME,
      t2.ORGANKEY,
      t2.ALERTDESC
FROM TB_XX_EXADATA t2,
     (SELECT distinct t.tempkey
      FROM TB_XX_EXADATA t
      GROUP BY t.tempkey
      HAVING count(t.tempkey) >= 10000) t1
WHERE t2.tempkey = t1.tempkey
```

执行计划

Plan hash value: 3759901922

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	INSERT STATEMENT					33G(100)	
1	LOAD TABLE CONVENTIONAL						
2	FILTER						
3	HASH GROUP BY		370T	607P		33G(100)	999:59:59
4	HASH JOIN		370T	607P	318M	1777M (83)	999:59:59
5	TABLE ACCESS STORAGE FULL	TB XX EXADATA	2932K	285M		30765 (1)	00:04:07
6	TABLE ACCESS STORAGE FULL	TB XX EXADATA	2650M	4307G		61191 (50)	00:08:10

原因和解决之道

原因，不用说了吧，刚刚讲完。本质是一个事
可以在inline view中加入no_merge的hint
进一步的可以用分析函数来优化，参考语法如下

```
select t2.TEMPKEY,  
       ...  
       from (SELECT t2.TEMPKEY,  
                 ...  
                 t2.ALERTDESC,  
                 count(*) over(partition by tempkey) count tempkey  
              FROM TB_XX_EXADATA t2) t2  
where count tempkey >= 10000
```

下一个案例



Perfecting IT service and favoring clients 'success
锻造凝练IT服务 助推用户事业发展

优化案例-执行计划选错索引和驱动表



Id	Operation	Name	Rows	Bytes	Cost	Time	Pstart
0	SELECT STATEMENT				13		
1	SORT GROUP BY		1	229			
2	FILTER						
3	NESTED LOOPS OUTER		1	229	13	00:00:01	
4	NESTED LOOPS		1	195	10	00:00:01	
5	NESTED LOOPS OUTER		1	159	8	00:00:01	
6	NESTED LOOPS OUTER		1	130	6	00:00:01	
7	TABLE ACCESS BY INDEX ROWID	TBL_CP_CREATE	1	101	4	00:00:01	
8	INDEX RANGE SCAN	CP_CREATETIME_IDX	1		3	00:00:01	
9	INDEX RANGE SCAN	CP_DRAFT_IDX	1	29	2	00:00:01	
10	INDEX RANGE SCAN	KFVISITDRAFTID	1	29	2	00:00:01	
11	TABLE ACCESS BY GLOBAL INDEX ROWID	TBL_CP_BILLSTATEMAP	1	36	2	00:00:01	ROW LOC
12	INDEX UNIQUE SCAN	SYS_C0034056	1		1	00:00:01	
13	TABLE ACCESS BY INDEX ROWID	TBL_CP_CREATE_USER	1	34	3	00:00:01	
14	INDEX RANGE SCAN	CP_CREATINFOID_IDX	1		2	00:00:01	

Predicate Information:

- 2 - filter(("TS_CREATE"."LOCATION"='0579' OR "T_USER"."LOCATION"='0579'))
- 8 - access("TS_CREATE"."CREATETIME">='2015-06-23 00:00:00' AND "TS_CREATE"."CREATETIME"<='2015-09-21 23:59:59')
- 9 - access("TS_CREATE"."DRAFTID"="ACCEPT"."DRAFTID")
- 10 - access("TS_KFBACK"."DRAFTID"="TS_CREATE"."DRAFTID")
- 11 - filter("TS_MAP"."BILLSTATE"='待报结')

问题在哪？如何解决？

1、看不出问题就是最大的问题

2、统计信息不对？

检查统计信息是相对准确的！

收集后问题依然！执行计划和COST和原来一样！

3、加hint或绑定执行计划？

找不到问题原因，如何去做到由点带面和预防类似问题呢？只能头痛医头！永远只能事后去绑定执行计划，而不能从根本上预防同类问题！

其实分析执行计划可以很快！

Id	Operation	Name	Rows	Bytes	Cost	Time	Pstart
0	SELECT STATEMENT				13		
1	SORT GROUP BY		1	229			
2	FILTER						
3	NESTED LOOPS OUTER		1	229	13	00:00:01	
4	NESTED LOOPS		1	195	10	00:00:01	
5	NESTED LOOPS OUTER		1	159	8	00:00:01	
6	NESTED LOOPS OUTER		1	130	6	00:00:01	
7	TABLE ACCESS BY INDEX ROWID	TBL_CP_CREATE	1	101	4	00:00:01	
8	INDEX RANGE SCAN	CP_CREATETIME_IDX	1		3	00:00:01	
9	INDEX RANGE SCAN	CP_DRAFT_IDX	1	29	2	00:00:01	
10	INDEX RANGE SCAN	KFVISITDRAFTID	1	29	2	00:00:01	
11	TABLE ACCESS BY GLOBAL INDEX ROWID	TBL_CP_BILLSTATEMAP	1	36	2	00:00:01	ROW LOC
12	INDEX UNIQUE SCAN	SYS_C0034056	1		1	00:00:01	
13	TABLE ACCESS BY INDEX ROWID	TBL_CP_CREATE_USER	1	34	3	00:00:01	
14	INDEX RANGE SCAN	CP_CREATINFOID_IDX	1		2	00:00:01	

Predicate Information:

```
2 - filter(("TS_CREATE"."LOCATION"='0579' OR "T_USER"."LOCATION"='0579'))
8 - access("TS_CREATE"."CREATETIME">='2015-06-23 00:00:00' AND "TS_CREATE"."CREATETIME"<='2015-09-21 23:59:59')
9 - access("TS_CREATE"."DRAFTID"="ACCEPT"."DRAFTID")
10 - access("TS_KFBACK"."DRAFTID"="TS_CREATE"."DRAFTID")
11 - filter("TS_MAP"."BILLSTATE"='待报结')
```

```
SQL> SELECT COUNT (*)  
2     FROM netforce.tbl_cp_create ts_create  
3     WHERE ts_create.CREATETIME >= '2015-06-23 00:00:  
4         AND ts_create.CREATETIME <= '2015-09-21 23:59:
```

COUNT (*)

358760

INDEX RANGE SCAN	CP_CREATETIME_IDX	1
INDEX RANGE SCAN	CP_DRAFT_IDX	1

我们来重现下问题

```
drop table t2;

create table t2(d1 date, c1 varchar2(20));

declare
  v date;
begin
  v := to_date('2008-12-04 22:43:38', 'yyyy-mm-dd hh24:mi:ss');
  for i in 1 .. 20000 loop
    insert into t2 values (v, to_char(v, 'yyyy-mm-dd hh24:mi:ss'));
    v := v + 0.3;
  end loop;
  commit;
end;
/
```

```
SQL> select min(c1),max(c1),min(d1),max(d1),count(*) from t1;
```

MIN(C1)	MAX(C1)	MIN(D1)	MAX(D1)	COUNT (*)
2008-12-04 22:43:38	2025-05-09 15:31:38	20081204 22:43:38	20250509 15:31:38	20000

```
SQL> select count(*) from t1  
       where c1 >= '2015-06-23 00:00:00'  
       and c1<='2015-09-21 23:59:59';
```

COUNT (*)
303

Varchar型c1的cardinality情况

```
SQL> explain plan for select * from t1 where c1 >= '2015-06-23 00:00:00' and c1<='2015-09-21 23:59:59';
```

Explained.

```
SQL> select * from table(dbms_xplan.display);
```

PLAN_TABLE_OUTPUT

Plan hash value: 3617692013

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		2	56	23 (5)	00:00:01
* 1	TABLE ACCESS FULL	T1	2	56	23 (5)	00:00:01

Predicate Information (identified by operation id):

```
1 - filter("C1"<='2015-09-21 23:59:59' AND "C1">='2015-06-23  
00:00:00')
```

Date型D1列的cardinality估算情况

```
SQL> explain plan for select * from t1
  where d1 between to_date('2015-06-23 00:00:00','yyyy-mm-dd hh24:mi:ss')
  and to_date('2015-09-21 23:59:59','yyyy-mm-dd hh24:mi:ss');
```

Explained.

```
SQL> select * from table(dbms_xplan.display);
PLAN_TABLE_OUTPUT
```

Plan hash value: 3617692013

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		305	8540	23 (5)	00:00:01
* 1	TABLE ACCESS FULL	T1	305	8540	23 (5)	00:00:01

Predicate Information (identified by operation id):

```
1 - filter("D1"<=TO_DATE('2015-09-21 23:59:59', 'yyyy-mm-dd
          hh24:mi:ss') AND "D1">=TO_DATE('2015-06-23 00:00:00', 'yyyy-mm-dd
          hh24:mi:ss'))
```

- 假设SQL语句为 `where col1 between :b1 and :b2`
- 本质原因在于 `where col1 between :b1 and :b2` 被当做了 `where col1 = :b1`, 即选择率为 $1/\text{num_distinct}$, 于是返回行数被错误的低估了
- 选择率简单公式 $\text{selectivity} = (\text{:b2} - \text{:b1}) / (\text{列最大值} - \text{列的最小值})$

续

```
SQL> select get_internal_value('2015-09-21 23:59:59') high,  
2 get_internal_value('2015-06-23 00:00:00') low from dual;
```

HIGH	LOW
<u>26059229722501500000000000000000000000</u>	<u>26059229722501500000000000000000000000</u>

CBO在计算字符串的>, <谓词的选择率时, 如果不存在直方图, 则需要把字符类型化成内部的数字类型, 然后按照 $selectivity = (:b2 - :b1) / (列最大值 - 列的最小值)$ 公式进行计算

我们可以看到, 由于:b2和:b1对应的' 2015-09-21 23:59:59', ' 2015-06-23 00:00:00' 这两个字符串的内部值是一样的, 所以按照公式, 分子为0, 评估出来的就非常低, 于是整个执行计划就错了!

结局方法、经验总结—由点带面

收集直方图即可！

需要说明的是，如果判断数据不倾斜，则默认不收集直方图，那么对于刚上线额系统是致命的！

不要采用varchar来存储date型数据！
仅此而已？还有其他的么？当做作业吧

总结和预防

- 1、通过根因分析，避免了长时间停机和数据可能不一致的问题
- 2、如何预防？ dataGuard /RAC ？



中亦科技吉祥物
海狸先生

THANK YOU

感谢您的关注