【故障处理】分布式事务 ORA-01591 错误解决

1.1 BLOG 文档结构图

1.2 前言部分

1.2.1 导读和注意事项

各位技术爱好者，看完本文后，你可以掌握如下的技能，也可以学到一些其它你所不知道的知识，~O(∩_∩)O~：

① 分布式事务的简单概念

② ORA-01591 错误解决

Tips：

① 本文在 ITpub（http://blog.itpub.net/26736162）、博客园（http://www.cnblogs.com/lhrbest）和微信公众号（xiaomaimiaolhr）有同步更新

② 文章中用到的所有代码，相关软件，相关资料请前往小麦苗的云盘下载（http://blog.itpub.net/26736162/viewspace-1624453/）

③ 若文章代码格式有错乱，推荐使用搜狗、360 或 QQ 浏览器，也可以下载 pdf 格式的文档来查看，pdf 文档下载地址：http://blog.itpub.net/26736162/viewspace-1624453/，另外 itpub 格式显示有问题，可以去博客园地址阅读

④ 本篇 BLOG 中命令的输出部分需要特别关注的地方我都用灰色背景和粉红色字体来表示，比如下边的例子中，thread 1 的最大归档日志号为 33，thread 2 的最大归档日志号为 43 是需要特别关注的地方；而命令一般使用黄色背景和红色字体标注；对代码或代码输出部分的注释一般采用蓝色字体表示。

```
  List of Archived Logs in backup set 11
  Thrd Seq     Low SCN    Low Time            Next SCN   Next Time
  ---- ------- ---------- ------------------- ---------- ---------
  1    32      1621589    2015-05-29 11:09:52 1625242    2015-05-29 11:15:48
  1    33      1625242    2015-05-29 11:15:48 1625293    2015-05-29 11:15:58
  2    42      1613951    2015-05-29 10:41:18 1625245    2015-05-29 11:15:49
  2    43      1625245    2015-05-29 11:15:49 1625253    2015-05-29 11:15:53

[ZHLHRDB1:root]:/>lsvg -o
T_XDESK_APP1_vg
rootvg
[ZHLHRDB1:root]:/>
00:27:22 SQL> alter tablespace idxtbs read write;

====》2097152*512/1024/1024/1024=1G
```

本文如有错误或不完善的地方请大家多多指正，ITPUB 留言或 QQ 皆可，您的批评指正是我写作的最大动力。

1.3 故障分析及解决过程

1.3.1 故障环境介绍

项目	source db
db 类型	RAC
db version	11.2.0.3

db 存储	ASM
OS 版本及 kernel 版本	AIX 64 位 6.1.0.0

1.3.2 故障发生现象及报错信息

有同事发来错误：

```
Select a.session_id,a.oracle_username,a.OS_USER_NAME,b.object_name
from v$locked_object a,user_objects b where a.object_id = b.object_id;

update TRGCBOKBAL set creditamt=creditamt+1.01 ,currbalance=currbalance+1.01  where bookaccount='1[illegible]' and currency='001';
```

执行一个 update 语句的时候报错 ORA-01591 的错误。

1.3.3 故障分析及解决过程

这个错误是由于分布式事务引起，而不是普通的锁引起的，检查一般对象数据表锁定，只需要检查 v$locked_object 和 v$transaction 视图，就可以定位到具体的 SQL 语句和操作人等信息，但是检查之后的结果如下：

```
SYS@oraLHR12>  select * from gv$locked_object;

no rows selected

SYS@oraLHR12> select * from gv$transaction;

no rows selected
```

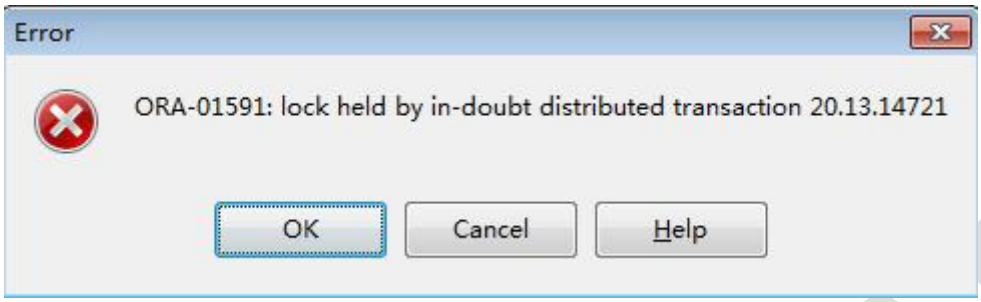
两个关键视图中，没有锁定的对象，也没有正在进行没有提交的事务。那是不是没有锁定呢？或者锁已经释放了，我们尝试对数据表加锁：

```
SYS@oraLHR12> select * from LHR.LHRBOKBAL for update;
select * from LHR.LHRBOKBAL for update
```

```
                *
ERROR at line 1:
ORA-01591: lock held by in-doubt distributed transaction 20.13.14721

SYS@oraLHR12> select count(1) from LHR.LHRBOKBAL;

  COUNT(1)
----------
  30998411
```



系统没有像一般阻塞那样等待，而是报错 ORA-01591 的错误，并且提示锁被一个分布式事务持有，不能实现加锁操作，那么 ORA-01591 错误究竟是什么呢？我们使用 oerr 工具查看该错误编号，看看有没有值得关注的信息。

```
root@ZFLHRRSP:/# oerr ora 1591
01591, 00000, "lock held by in-doubt distributed transaction %s"
// *Cause:  Trying to access resource that is locked by a dead two-phase commit
//         transaction that is in prepared state.
// *Action: DBA should query the pending_trans$ and related tables, and attempt
//         to repair network connection(s) to coordinator and commit point.
//         If timely repair is not possible, DBA should contact DBA at commit
//         point if known or end user for correct outcome, or use heuristic
//         default if given to issue a heuristic commit or abort command to
//         finalize the local portion of the distributed transaction.
```

简单的说，01591 错误的原因是该对象被一个处在“in-doubt”状态的分布式事务锁定。分布式事务使用的是“two-phase commit”二阶段提交技术。解决该问题的方法就是查看内部表 pending_trans$，确定分布式事务信息。这种状态的事务主要是由于在进行分布式事务时候，发生网络突发中断的情况，引起分布式事务无法正常结束，等待中断节点的事务响应。于是，各节点的事务所锁定的表就不会被释放掉。

此时，我们检查视图 DBA_2PC_PENDING（或者基表 pending_trans$），查看是否存在这种情况。

Row 1	Fields	Comments
LOCAL_TRAN_ID	20.13.14721	*string of form: n.n.n, n a number*
GLOBAL_TRAN_ID	113577.6CC9A6EC98F6310D082D19ED28B9F3E62662821F	*globally unique transaction id*
STATE	prepared	*collecting, prepared, committed, forced commit, or forced rollback*
MIXED	no	*yes => part of the transaction committed and part rolled back (cor*
ADVICE		*C for commit, R for rollback, else null*
TRAN_COMMENT		*text for "commit work comment <text>"*
FAIL_TIME	2016/8/3 18:11:49	*value of SYSDATE when the row was inserted (tx or system recov*
FORCE_TIME		*time of manual force decision (null if not forced locally)*
RETRY_TIME	2016/8/4 8:10:37	*time automatic recovery (RECO) last tried to recover the transacti*
OS_USER	[redacted]	*operating system specific name for the end-user*
OS_TERMINAL		*operating system specific name for the end-user terminal*
HOST	[redacted]	*name of the host machine for the end-user*
DB_USER		*Oracle user name of the end-user at the topmost database*
COMMIT#	46784414	*global commit number for committed transactions*

果然，当前存在一个阻塞分布式事务，处在 prepared 状态。当前问题，主要是源于在进入 prepared 阶段之

后，发生了网络中断的现象，引起 commit 的阶段不能等待到事务信息。所以，才会一直处在 Prepared 状态，数据表也就不会进行释放。

对于这个事务，只能通过连接网络或者强制提交回退事务来结束。我们可以使用 commit force 或者 rollback force 来进行处理，这里我们进行回滚操作：

```
SYS@oraLHR12> rollback force '20.13.14721';

Rollback complete.

SYS@oraLHR12>
```

Rollback force 的参数是 DBA_2PC_PENDING 中记录本地事务信息的编号即 LOCAL_TRAN_ID。

此时，再次查看数据。

Row 1	Fields	Comments
LOCAL_TRAN_ID	20.13.14721	string of form: n.n.n, n a number
GLOBAL_TRAN_ID	113577.6CC9A6EC98F6310D082D19ED28B9F3E62662821F	globally unique transaction id
STATE	forced rollback	collecting, prepared, committed, forced commit, or forced rollback
MIXED	no	yes => part of the transaction committed and part rolled back (co
ADVICE		C for commit, R for rollback, else null
TRAN_COMMENT		text for "commit work comment <text>"
FAIL_TIME	2016/8/3 18:11:49	value of SYSDATE when the row was inserted (tx or system recov
FORCE_TIME	2016/8/4 15:10:24	time of manual force decision (null if not forced locally)
RETRY_TIME	2016/8/4 15:10:43	time automatic recovery (RECO) last tried to recover the transacti
OS_USER	[redacted]	operating system specific name for the end-user
OS_TERMINAL		operating system specific name for the end-user terminal
HOST	[redacted]	name of the host machine for the end-user
DB_USER		Oracle user name of the end-user at the topmost database
COMMIT#	46784414	global commit number for committed transactions

此时，该事务状态已经变化为 forced rollback 表示已经强制回退，我们再次尝试锁定表操作：

```
16:25:31 SQL> select CURRENCY from tpcc.TPCCBOKBAL WHERE ROWNUM=1  for update;
CURRENCY
--------
001
Executed in 0.025 seconds
```

可以看出已经不报错了，可以正常执行。

1.4 分布式事务相关知识点

分布式事务，简单来说，是指一个事务在本地和远程执行，本地需要等待确认远程的事务结束后，进行下一步本地的操作。如通过 dblink update 远程数据库的一行记录，如果在执行过程中网络异常，或者其他事件导致本地数据库无法得知远程数据库的执行情况，此时就会发生 in doublt 的报错。此时需要 dba 介入，且需要分多种情况进行处理。

Oracle 会自动处理分布事务，保证分布事务的一致性，所有站点全部提交或全部回滚。一般情况下，处理过程在很短的时间内完成，根本无法察觉到。

但是，如果在 commit 或 rollback 的时候，出现了连接中断或某个数据库 站点 CRASH 的情况，则提交操作可能会无法继续，此时 DBA_2PC_PENDING 和 DBA_2PC_NEIGHBORS 中会包含尚未解决的分布事务。 对于绝大多数情况，当恢复连接或 CRASH 的数据库重新启动后，会自动解决分布式事务，不需要人工干预。只有分布事务锁住的对象急需被访问，锁住的回滚段阻止了其他事务的使用，网络故障或 CRASH 的数据库的恢复需要很长的时间等情况出现时，才使用人工操作的方式来维护分布式事务。 手工强制提交或回滚将失去二层提交的特性，Oracle 无法继续保证事务的一致性，事务的一致性应由手工操作者保证

使用 ALTER SYSTEM DISABLE DISTRIBUTED RECOVERY，可以使 Oracle 不再自动解决分布事务，即使网络恢复连接或者 CRASH 的数据库重新启动。

ALTER SYSTEM ENABLE DISTRIBUTED RECOVERY 恢复自动解决分布事务。

1.5 两个重要的视图

1.5.1 DBA_2PC_PENDING

DBA_2PC_PENDING:列出所有的悬而未决的事务，此视图在未填入悬而未决的事务之前是空的，解决这后也被清空。

列名	说明
LOCAL_TRAN_ID	本地事务标识，格式为 integer.integer.ingeger。 当一个连接的 local_tran_id 和 global_tran_id 相同时，那么该节点是该事务的全局协调器。
GLOBAL_TRAN_ID	全局事务标识,格式为： global_db_name.db_hex_id.local_tran_id,其中 db_hex_id 是用来标识数据库八字符的十六进制数，公共事各 id 在分布式事务的每个节点都是相同的。
STATE	下图表进行说明
MIXED	“YES”意味着一部分事务已经在一个节点上提交，而在另一个节点上被回滚。
TRAN_COMMENT	事务的注释，或者如果使用了事务命名，当事各被提交时，事务的名字就会

	出现在此处
Host	主机名
Commit#	已提交的事务的全局提交数

DBA_2PC_PENDING 的 STATE 列的说明

列值	说明
Connecting	通常情况下，只有全局协调器和本地协调器才使用这个条目，节点在能够决定它是否能够准备好之前，要收集来自于其它数据库服务的信息。
Prepared	节点已准好，可能或者也可能没有将已准备好的消息通知本地协调器，但此时，该节点还没有接收到提交的请求，仍保持着准许备好的状态，控制着提交事务所必需的任何本地资源。
Commited	节点(任何类型)已经提交了事务，但该事务所包含的其它节点可能并没有提交，也就是该事务在一个个或多个其它节点上仍然是悬而未决 。
Forced commit	DBA 进行判断后，可以强行提交未决的事务，如果一个事务由 DBA 在本地节点进行手动提交时，产生此项目
Forced abor(rollback)	DBA 进行判断后，可以强行回滚未决的事务，如果一个事务由 DBA 在本地节点进行手动回滚时，产生此项目

```
SELECT * FROM DBA_2PC_PENDING;
```

LOCAL_TRAN_ID	GLOBAL_TRAN_ID	STATE	MIXED	ADVICE	TRAN_COMMENT	FAIL_TIME
89.20.111	113577.1BD60DA5AC737E4AECFBFDF7658959F26CD4826C	prepared	no			2016/6/14 16:13:54
171.25.152	113577.2D3EC176242BEE11ADE3D81C474606039C1BCDEB	prepared	no			2016/6/14 16:13:54
172.31.98	113577.690A708EA337B4A99DC94E82179F9544EBA6D355	prepared	no			2016/6/14 16:13:54
181.24.148	113577.C8F3CA10F4104BBB35938F732C0DE3F8ACB6C300	prepared	no			2016/6/14 16:13:54
122.26.155	113577.01F0992D3691475E246EF885AC542ED9A64C87D1	prepared	no			2016/6/14 16:22:40
291.11.266	113577.375BEFD7B0977A659BC4A71813FDFAF4F4BB9E0B	prepared	no			2016/6/16 14:43:53

1.5.2 DBA_2PC_NEIGHBORS

DBA_2PC_NEIGHBORS:列出所有获得的(从远程客户)和送出的(给远程服务器)悬而未决的事务，也表示该本地节点是不是事务的提交点站点。

列名	说明
LOCAL_TRAN_ID	同上
IN_OUT	获得事务为 IN，送出事务为 OUT
Database	对获得事务来说指本地节点信息的客户数据库的名称；对送出的事务来说指用于访问远程服务器上信息的数据库链接的名称
DBuser_owner	对获得事务来说指远程数据库链接用于连接的本地账户；对于送出事务来说指该数据库链接的拥有者。

INTERFACE	‘C’代表提交信息，’N’表示已准备好状态的一条消息或是一条请求只读提交的请求。 当’IN_OUT’为 OUT 时，’C’表示该连接的远程的站点是提交点站点，并且知道是提交还是中断。’N’表示本地节点正在通知远程节点，说它已准备好。 当’IN_OUT’为 IN 时，‘C’表示本地节点或送出的远程的一个数据库是提交点站点，’N’表示本地节点正在通知远程节点，说它已准备好。

About Me

长按识别二维码或微信客户端扫描下边的二维码来关注小麦苗的微信公众号：xiaomaimiaolhr，学习最实用的数据库技术。


小麦苗