- QQ群1：5276420
- QQ群2：3336901
- QQ群3：254622631
- 文档群：150657323
- 文档翻译平台：按此
- 邮件列表：按此订阅

官方微信公众号

官方微博

搜索：

用户：

密码：

登录 注册

晋级

PostgreSQL的企业级云服务

PostgreSQL携手阿里云

PostgreSQL流复制案例分析 | Startup 进程waiting问题分析

原作者：刘雪柏 创作时间：2019-08-12 10:41:13+08

wangliyun 发布于2019-08-13 08:08:13 评论: 1 浏览: 2418 顶: 176 踩: 16

一、 作者介绍

刘雪柏，目前就职于诺基亚东软通信有限公司，高级软件工程师，从事数据库相关工作，联系方式: **aiden.liu@qq.com**。

二、 问题现象

Postgresql 11.2版本物理复制，startup 进程命令行有时会出现waiting 标识。本文分析了出现waiting 标识的原因。

三、 问题分析:

通过ps -ef | grep startup 监控startup 进程。

postgres: startup process recovering 0000000100000000000000A6

当startup 进程出现waiting的标识。通过pstack查看startup 进程调用堆栈

[postgres@sscloud21 ~]$ pstack 197120

#0 0x00007f695b6faf53 in _*select*nocancel () from /lib64/libc.so.6

#1 0x000000000088c97a in pg_usleep (microsec=) at pgsleep.c:56

#2 0x0000000000729ef9 in WaitExceedsMaxStandbyDelay () at standby.c:201

#3 ResolveRecoveryConflictWithVirtualXIDs (waitlist=0x1c706e0, reason=reason@entry=PROCSIG*RECOVERY*CONFLICT_SN at standby.c:262

#4 0x000000000072a10e in ResolveRecoveryConflictWithVirtualXIDs (reason=PROCSIG*RECOVERY*CONFLICT_SNAPSHOT, w standby.c:315

#5 ResolveRecoveryConflictWithSnapshot (latestRemovedXid=, node=…) at standby.c:313

#6 0x00000000004c23be in heap*xlog*clean (record=0x1c00698) at heapam.c:8198

#7 heap2_redo (record=0x1c00698) at heapam.c:9351

#8 0x0000000000503e85 in StartupXLOG () at xlog.c:7306

#9 0x00000000006d82b1 in StartupProcessMain () at startup.c:211

#10 0x0000000000512275 in AuxiliaryProcessMain (argc=argc@entry=2, argv=argv@entry=0x7fff8b5d99b0) at bootstrap.c:441

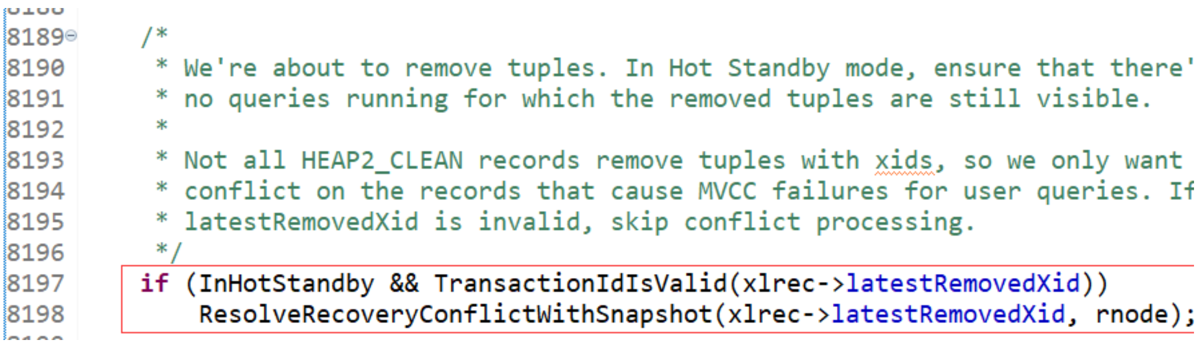
#11 0x00000000006d53a0 in StartChildProcess (type=StartupProcess) at postmaster.c:5331

#12 0x00000000006d7b75 in PostmasterMain (argc=argc@entry=3, argv=argv@entry=0x1bd0e40) at postmaster.c:1371

#13 0x000000000048124f in main (argc=3, argv=0x1bd0e40) at main.c:228

检查源码发现，**startup** 进程正在执行 **XLOG*HEAP2*CLEAN** 类型的**redo**操作（**vacuum**命令产生的块清除**redo**）。

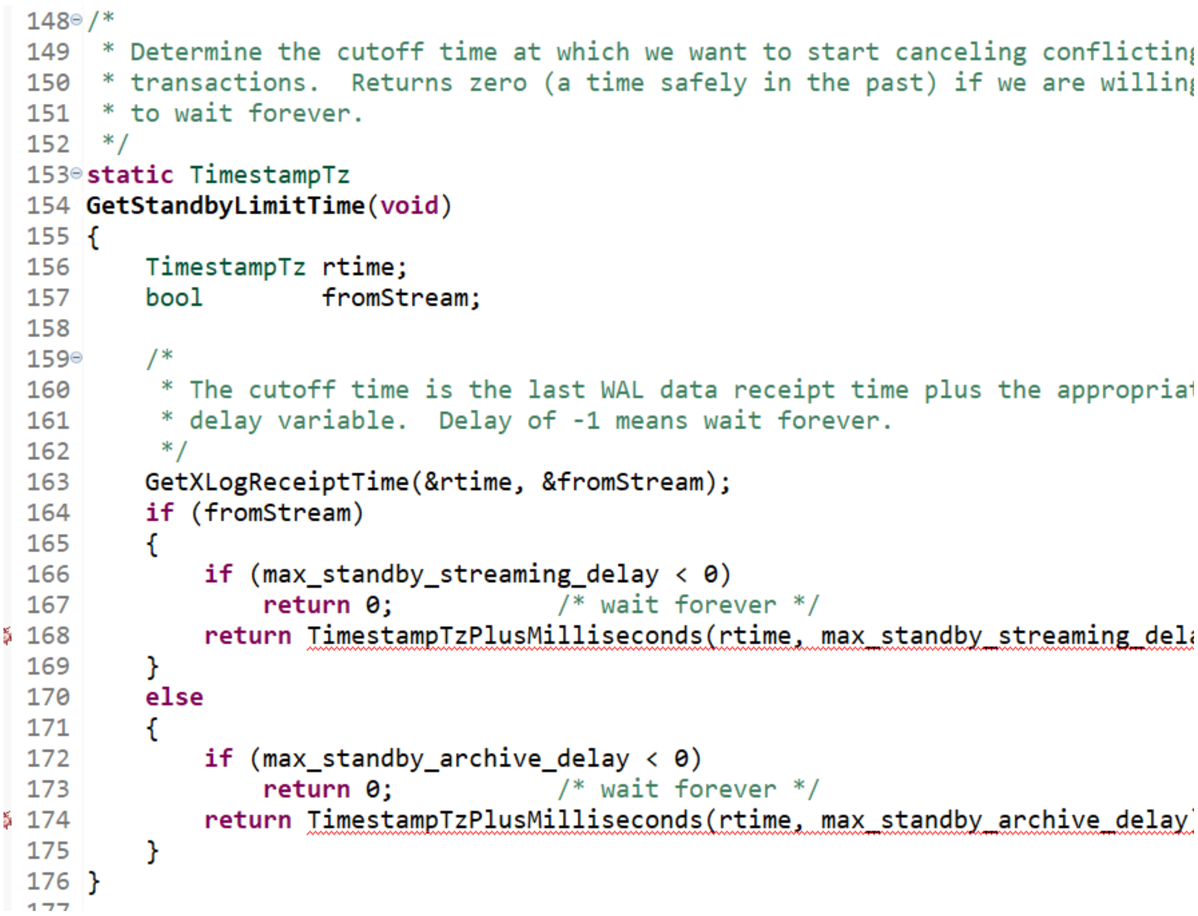
heapam.c代码中有如下注释：


```
    /*
     * We're about to remove tuples. In Hot Standby mode, ensure that there'
     * no queries running for which the removed tuples are still visible.
     *
     * Not all HEAP2_CLEAN records remove tuples with xids, so we only want
     * conflict on the records that cause MVCC failures for user queries. If
     * latestRemovedXid is invalid, skip conflict processing.
     */
    if (InHotStandby && TransactionIdIsValid(xlrec->latestRemovedXid))
        ResolveRecoveryConflictWithSnapshot(xlrec->latestRemovedXid, rnode);
```

startup 进程在执行清除**tuples**前需要确认没有活动的查询，并且这些**tuples**对于活动查询仍然可见。

跟着**ResolveRecoveryConflictWithSnapshot->ResolveRecoveryConflictWithVirtualXIDs->WaitExceedsMaxStandbyDelay->GetStandbyLimitTime**

standby.c代码中**GetStandbyLimitTime**函数：

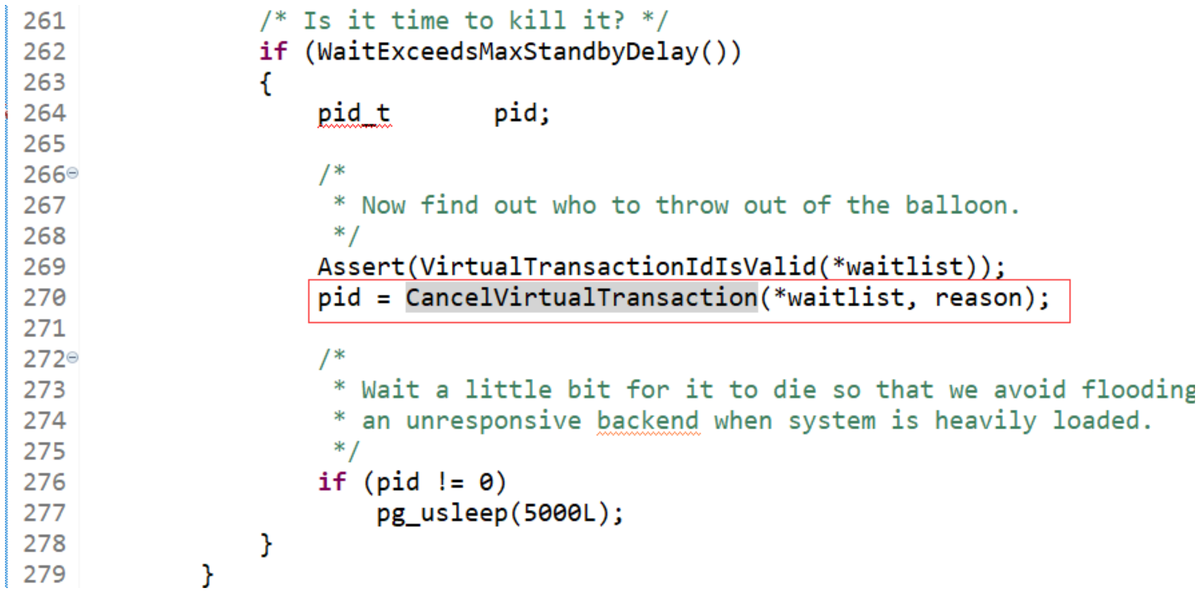

```
/*
 * Determine the cutoff time at which we want to start canceling conflicting
 * transactions.  Returns zero (a time safely in the past) if we are willin
 * to wait forever.
 */
static TimestampTz
GetStandbyLimitTime(void)
{
    TimestampTz rtime;
    bool        fromStream;

    /*
     * The cutoff time is the last WAL data receipt time plus the appropria
     * delay variable.  Delay of -1 means wait forever.
     */
    GetXLogReceiptTime(&rtime, &fromStream);
    if (fromStream)
    {
        if (max_standby_streaming_delay < 0)
            return 0;           /* wait forever */
        return TimestampTzPlusMilliseconds(rtime, max_standby_streaming_del
    }
    else
    {
        if (max_standby_archive_delay < 0)
            return 0;           /* wait forever */
        return TimestampTzPlusMilliseconds(rtime, max_standby_archive_delay
    }
}
```

从代码中可以看出**startup** 进程在执行清除**tuples**前需要确认没有活动的查询有等待时间限制，**fromStream**的情况超过

max*standby***streaming***delay参数时间限制会kill掉正在执行的查询，其他情况如果超过了max***standby***archive***delay**

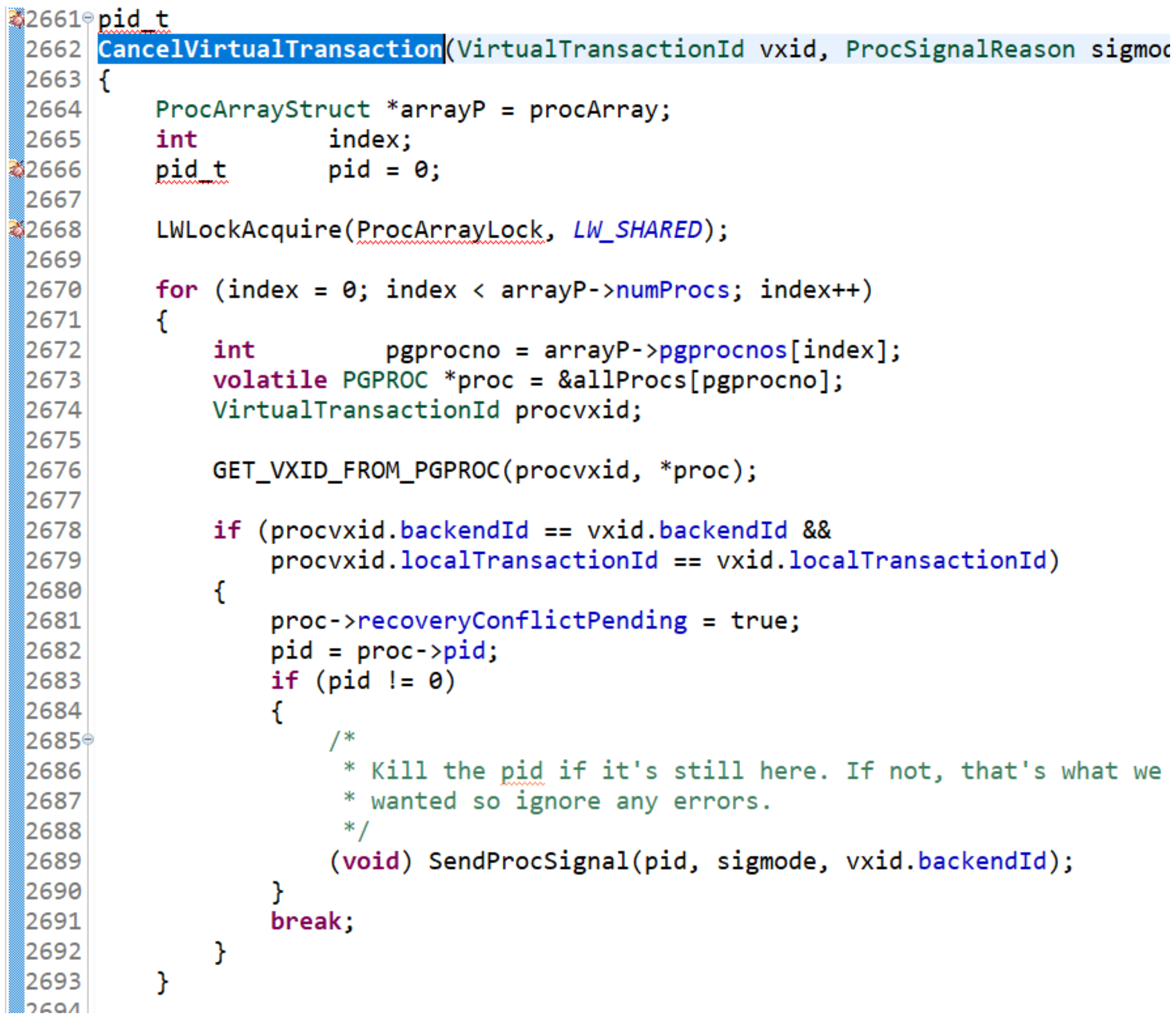
参数时间限制会调用**CancelVirtualTransaction**函数 **kill**掉正在执行的查询。

standby.c相关代码



```
            /* Is it time to kill it? */
            if (WaitExceedsMaxStandbyDelay())
            {
                pid_t       pid;

                /*
                 * Now find out who to throw out of the balloon.
                 */
                Assert(VirtualTransactionIdIsValid(*waitlist));
                pid = CancelVirtualTransaction(*waitlist, reason);

                /*
                 * Wait a little bit for it to die so that we avoid flooding
                 * an unresponsive backend when system is heavily loaded.
                 */
                if (pid != 0)
                    pg_usleep(5000L);
            }
        }
```

procarray.c相关代码



```
pid_t
CancelVirtualTransaction(VirtualTransactionId vxid, ProcSignalReason sigmo
{
    ProcArrayStruct *arrayP = procArray;
    int         index;
    pid_t       pid = 0;

    LWLockAcquire(ProcArrayLock, LW_SHARED);

    for (index = 0; index < arrayP->numProcs; index++)
    {
        int         pgprocno = arrayP->pgprocnos[index];
        volatile PGPROC *proc = &allProcs[pgprocno];
        VirtualTransactionId procvxid;

        GET_VXID_FROM_PGPROC(procvxid, *proc);

        if (procvxid.backendId == vxid.backendId &&
            procvxid.localTransactionId == vxid.localTransactionId)
        {
            proc->recoveryConflictPending = true;
            pid = proc->pid;
            if (pid != 0)
            {
                /*
                 * Kill the pid if it's still here. If not, that's what we
                 * wanted so ignore any errors.
                 */
                (void) SendProcSignal(pid, sigmode, vxid.backendId);
            }
            break;
        }
    }
```

四、 小结

1、 原因: 重放XLOG*HEAP2*CLEAN 类型的redo操作不允许有查询，如果有会等待，等待时间由max*standby*streaming*delay/max*standby*archive*delay控制。

2、 解决这个问题可以控制slave端禁止执行查询操作，或者调整max*standby*streaming*delay/max*standby*archive*delay参数的值到范围。

3、 max*standby*streaming*delay/max*standby*archive*delay参数默认值30秒。

评论:1 浏览: 2418 顶: 176 踩: 162

评论： 请在登录后发表评论，否则无法保存。

1# __ hemuyang 回答于 2019-09-15 10:54:23+08
请问下如何复现该问题

发表评论：

您还没有登录,请您登录后再发表评论

Debug: Elapsed Time: 0.0214s / Memory Used: 2.01MB