为什么hot\_standby\_feedback可能会产生误导

作者：RICHARD YEN

翻译：陈雁飞

## 引言

当我第一次参与Postgres数据库管理的时候，很快就了解了复制的必要性。我的第一个项目是将数据库放在Slony上，这是一种热门的新复制技术，代替笨拙的DRBD配置，并且支持对数据库近实时的副本备份。当然，随着时间和规模的增加，Slony很难满足快速增加的写入流量，并且最终会受到写放大的影响（由于所有操作包含对底层操作，因此每次操作最终会变成对数据库的两次或者更多次写）。当在Postgres9.0版本出现流复制的时候，每个人都觉得自己得到了金牌。流复制不仅速度快，而且它利用了Postgres中已经存在的特性：WAL stream。

距离Postgres9.0版本已经过去很多年了（我们马上见到12版本）。这期间Postgres增加了很多特性，比如热备份、逻辑复制和一些双向主-主复制扩展插件。这个是一条不寻常的成长之路，特别是我记得在大约是在2010年的PGcon有人介绍BOF的时候说，Postgres的路线图中将不会包括复制。

在多年来流复制的所有改进中，我认为一个最容易被误解特性是hot\_standby\_feedback，因此我希望在这里能够澄清一下。

通过流复制，用户可以建立任意个与主机有克隆复制关系的备服务器，并且这些备服务器也支持一些特定类型的操作。比如OLTP应用程序中只读流量、庞大的定时任务、以及长时间运行的报表查询等，所有这些操作不会影响主服务器上的写入流量。但是，有些人偶尔会遇到查询由于某种原因被终止了，并且在日志中，可以看到类似如下内容：

|  |
| --- |
| ERROR: canceling statement due to conflict with recovery |

这个错误是一个不幸的现实，没有人会喜欢这个。没有人希望自己的查询被取消，就像没人喜欢在点一个熏牛肉三明治的时候，过了10分钟才被店家告知熏牛肉卖完了。但是，这种情况确实发生了。通常在流复制中一些数据需要重放的时候发生[查询冲突](https://www.postgresql.org/docs/current/hot-standby.html#HOT-STANDBY-CONFLICT)的情况，常见的是一些旧数据的删除，无论是删除，还是删除表，甚至VACUUM清理死行数据操作。在Google上搜索Postgres流复制的时候，一些用户建议通过设置hot\_standby\_feedback参数为on的方式解决遇到的查询冲突问题。很多人很高兴地设置hot\_standby\_feedback = on，然后一切就重新开始工作了，但是我遇到一些客户设置之后仍然会从日志中看到ERROR: canceling statement due to conflict with recovery错误信息。

## 为什么继续产生查询冲突？

关于Postgres的流复制，需要牢记的重要一点：它的目标是创建数据库副本，并将WAL日志传输到另外一边去。现在，DBA可能有其他的不同目标，包括在备服务器上进行查询和报表操作，但是这个并不是Postgres的目标。DBA可以配置一些GUC参数，用于告诉Postgres在某些方面不需要太激进，以便能预留一点空间在备机上运行查询操作。但是，如果备机没有处理接收到的WAL流文件，那么在pg\_xlog/pg\_wal目录下的WAL日志将堆积并膨胀，并且使运行的主服务器面临磁盘空间耗尽的风险。

## hot\_standbt\_feedback究竟是做什么的？

这里不想再谈技术细节了，我认为[Alexey Lesovsky](https://blog.dataegret.com/2015/09/postgresql-hot-standby-feedback-how-it.html)用一个很好的例子解释了它，并且有源代码提供给有兴趣的人详细阅读。简而言之，改参数表示备机将相关信息发送给主机（pg\_stat\_replication.backend\_xmin值），从而帮助主机确定哪些dead tuple可以安全地被vacuum清理掉。换句话说，当备机上执行查询的时候，通过特定的backend\_xmin值确定可见性，从而主机不会vacuum清理相关元组。

 -bash-4.1$ psql -p5432 -c "SELECT application\_name, backend\_start, backend\_xmin, state, sent\_location, write\_location, flush\_location, replay\_location FROM pg\_stat\_replication"  
  application\_name |         backend\_start         |     backend\_xmin     |   state   | sent\_location | write\_location | flush\_location | replay\_location  
 ------------------+---------------------------------+-----------------------+-----------+---------------+----------------+----------------+-----------------  
  walreceiver     | 01-MAR-19 23:18:55.31685 +00:00 |{look ma, nothing here}| streaming | 0/6000060     | 0/6000060     | 0/6000060     | 0/6000060  
 (1 row)  
 -bash-4.1$ psql -p5433 -c "ALTER SYSTEM SET hot\_standby\_feedback TO on"  
 ALTER SYSTEM  
 -bash-4.1$ pg\_ctl -D /var/lib/pgsql/9.6/standby\_pgdata restart  
 waiting for server to shut down.... done  
 server stopped  
 server starting  
 -bash-4.1$ psql -p5432 -c "SELECT application\_name, backend\_start, backend\_xmin, state, sent\_location, write\_location, flush\_location, replay\_location FROM pg\_stat\_replication"  
  application\_name |         backend\_start           |     backend\_xmin     |   state   | sent\_location | write\_location | flush\_location | replay\_location  
 ------------------+----------------------------------+-----------------------+-----------+---------------+----------------+----------------+-----------------  
  walreceiver     | 01-MAR-19 23:21:18.373624 +00:00 | {hs\_feedback on} 2346 | streaming | 0/6000060     | 0/6000060     | 0/6000060     | 0/6000060  
 (1 row)

基本上，这是一个延迟策略，将清除信息排除在WAL之外，因为一旦deaded元组被vacuum清理，清理信息将写入WAL日志流中并在备机上产生冲突。注意这里有一点需要权衡考虑：设置hot\_standby\_feedback=on可能会导致主机上表发生膨胀，通常这个并不是很重要。但是，即使这样，仍然有一些查询冲突是hot\_standby\_feedback无法阻止的：

1、对主机对象上请求排他锁

2、断断续续地walreceiver连接进程

3、对少数表的频繁修改

### 排它锁导致的冲突

为了满足流复制的可靠性，所有的备机需要与主服务器保持一致。这也就是说主机上所有的修改需要尽可能快发送到备机并恢复，这里需要特别注意的是DDL操作以及其他需要获取排它锁的变化。任何时延都可能导致数据库的不一致（谁会愿意看到已经删除的表？），且当在主机发生故障并且需要立即进行故障转移的时候，最理想的情况是备机与主机之间是完全同步的。因此，DBA希望能尽可能快的重放WAL日志。

一致性还意味着，当Alice和Bob在只读的备机上查询得到的结果是准确的，和主机上运行查询结果一样（当然，除非DBA已经将备机设置在固定时间内跟踪主机，比如在recovery.conf中设置recovery\_min\_apply\_delay参数）。但是，如果有人在数据库里执行一个长时间运行的查询，那么这个查询会阻止WAL日志的重放，因此必须给出一些参数用于取消这些查询操作。DBA可以设置max\_standby\_archive\_delay或max\_standby\_streaming\_delay参数，以便在取消查询前给冲突查询一段时间。这两个参数在设置的要谨慎，需要考虑到业务允许存在的复制时延。

### 断断续续的walreceiver连接的影响

网络中断将断开walreceiver连接，并最终使得发送给主机的backend\_xmin信息失效。也就是说，如果walreceiver连接断开，那么在walreceiver重新连接并告诉主机新的backend\_xmin信息之前，主机可以自由地对期望的对象进行vacuum清理。从备机用户角度看，一些非期望的vacuum清理操作可能发生在walreceiver进程重连期间，从而可能导致程出现查询冲突取消的情况。这种情况可以通过流复制槽来缓解，它可以记录断开walreceiver连接的xmin信息。

### 少量表的频繁写入

VACUUM操作通常不是阻塞的，但是如果存在有足够的查询/删除操作，一个vacuum任务可能会发现一个标记为需要删除的完整页面，在这种情况它将尝试获取对象的排它锁信息，从而将该页面从磁盘上删除，从而减小表占用的空间。从流复制恢复角度看，这种锁行为基本上与DDL操作一样，并且最终可能会导致正在运行中查询被取消掉。

## 总结

即使设置了hot\_standby\_feedback=on，仍然有很多种情况会导致查询因为冲突而取消，但是这些情况比较少见。查询pg\_stat\_database\_conflicts信息有助于了解产生取消查询的原因，对找到缓解该问题的方法有很大帮助。最后，hot\_standby\_feedback只是众多处理备机查询冲突方法中的一种，用户需要明白的一点是：为了保证一致和可靠，这种查询冲突时必须的。无论是通过hot\_feed\_back参数还是其他方式，设置备机查询为更高优先级，都是以牺牲主备之间复制时延（有时可能会很小）为代价。

原文地址：

<https://www.enterprisedb.com/blog/why-hotstandbyfeedback-can-be-misleading>