PostgreSQL

第12届PostgreSQL中国技术大会

安全可靠 × 突破 × 进化

逻辑复制原理与实践经验分享

施博文 腾讯云数据库 内核开发工程师

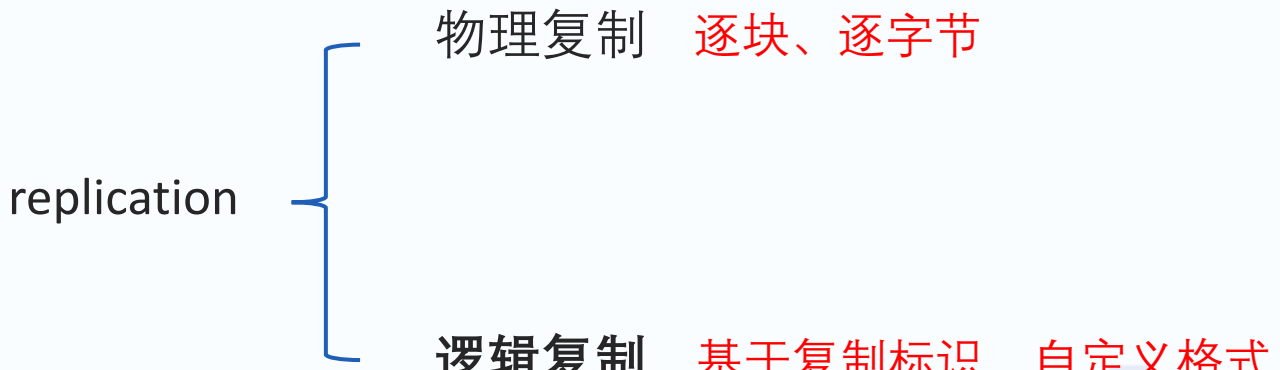
目录 Contents

01 逻辑复制概念和原理

逻辑复制概念和原理

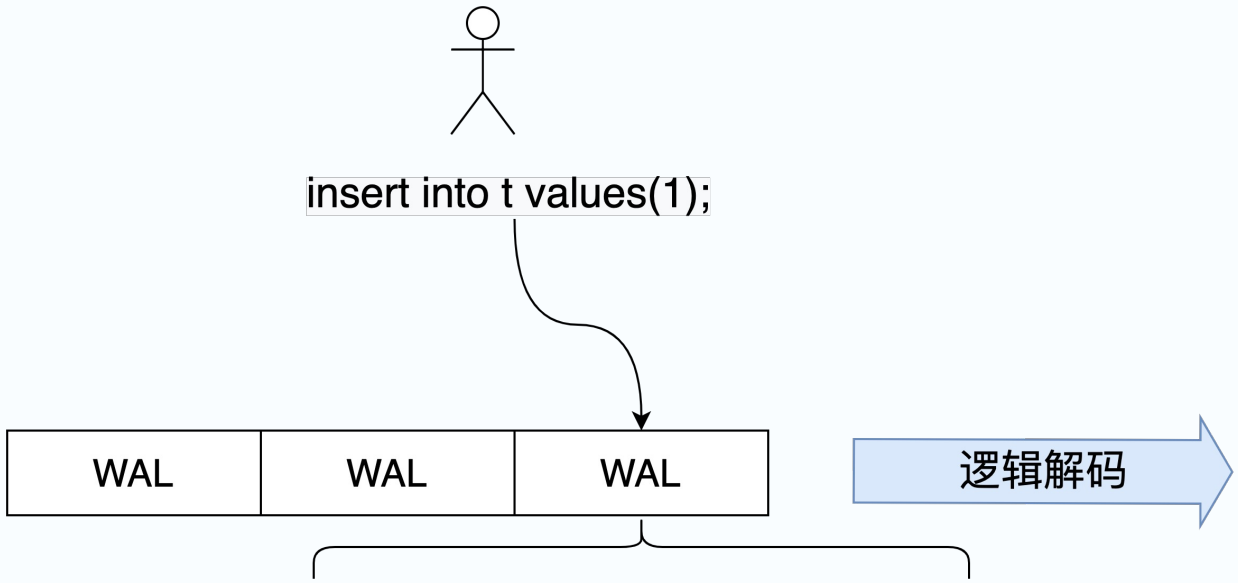
逻辑复制是什么？

- Logical replication is a method of replicating data objects and their changes, based upon their replication identity (usually a primary key). We use the term logical in contrast to physical replication, which uses exact block addresses and byte–by–byte replication.



逻辑复制概念和原理

物理日志 vs 逻辑日志



逻辑解码

```
postgres=# select pg_logical_slot_get_changes('dr',NULL,NULL);
              pg_logical_slot_get_changes
-----------------------------------------------------
 (0/17968F0,739,"INSERT INTO public.t (a) VALUES (1);")
```

逻辑日志：取决于解码插件。自定义格式

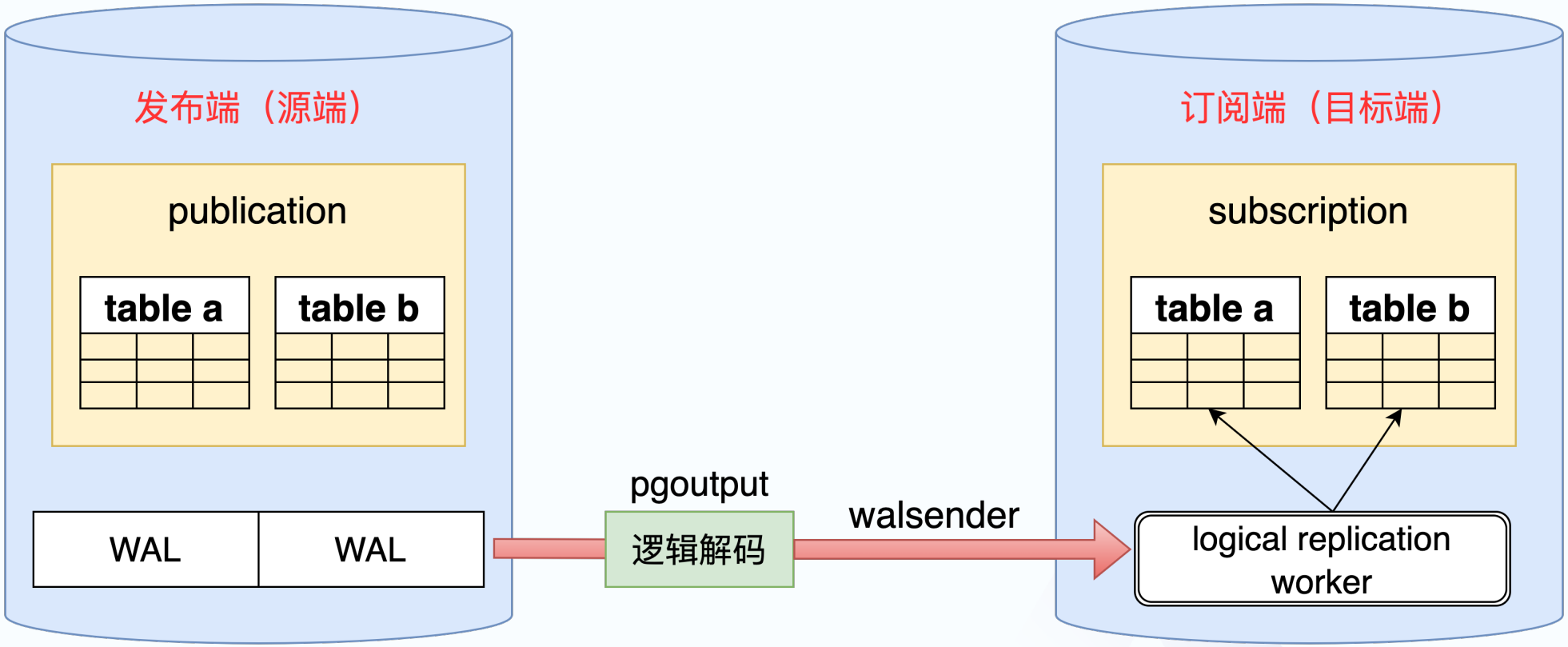
```
rmgr: Heap        len (rec/tot): 59/59, tx: 7621,
lsn: 0/0A0CF638, prev 0/0A0CF608,
desc: INSERT off 5 flags 0x08, blkref #0: rel 1663/14486/16384 blk 0
```

物理日志：记录对某个文件某个块的修改

逻辑复制概念和原理

发布-订阅

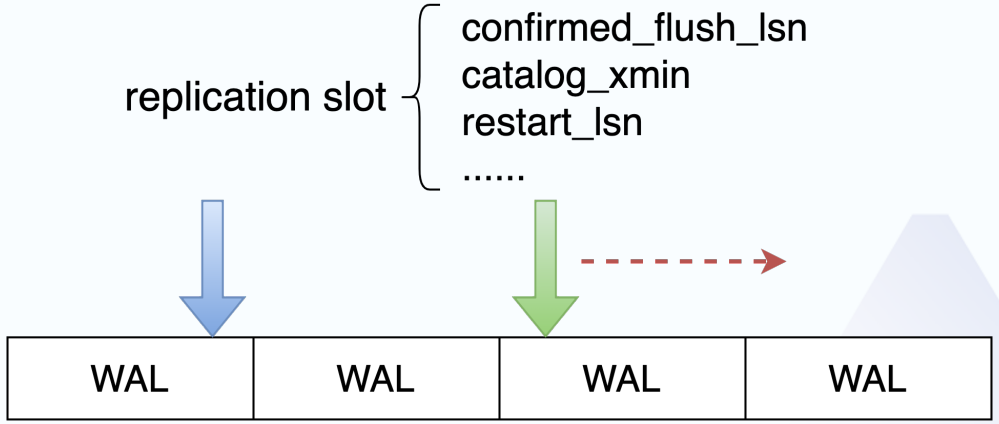
使用逻辑复制完成数据的同步



逻辑复制概念和原理

Replication Slot 复制槽

- 记录逻辑复制位点信息，保留需要的数据
  - confirmed_flush_lsn：当前逻辑复制已经同步到的 WAL LSN 位点
  - restart_lsn：逻辑复制需要的最小 LSN，避免该 LSN 之后的 WAL 被清理掉
  - catalog_xmin：当前需要的最小的 xid，只有小于该 xid 的系统表元组才可以被 vacuum 掉
  - ……



02

逻辑复制实践经验分享

逻辑复制实践经验分享

- PG 实例重启卡住

- 逻辑复制 walsender 进程卡住

逻辑复制实践经验分享

PG 实例重启卡住

- **背景**：用户重启时，实例运行 pg_ctl stop 命令卡住，无法停库
- **现场**：数据库日志无异常，现场堆栈可分为以下三种：

walsender 进程

```
#0  CheckXLogRemoved ()
#1  XLogRead ()
#2  logical_read_xlog_page ()
#3  ReadPageInternal ()
#4  XLogReadRecord ()
#5  XLogSendLogical ()
#6  WalSndLoop ()
#7  exec_replication_command ()
#8  PostgresMain ()
#9  ServerLoop ()
#10 PostmasterMain ()
#11 main ()
```

checkpointer 进程

```
#0  __select_nocancel ()
#1  pg_usleep ()
#2  WalSndWaitStopping ()
#3  ShutdownXLOG ()
#4  CheckpointerMain ()
#5  AuxiliaryProcessMain ()
#6  StartChildProcess ()
#7  sigusr1_handler ()
#8  <signal handler called>
#9  __select_nocancel ()
#10 ServerLoop ()
#11 PostmasterMain ()
#12 main ()
```

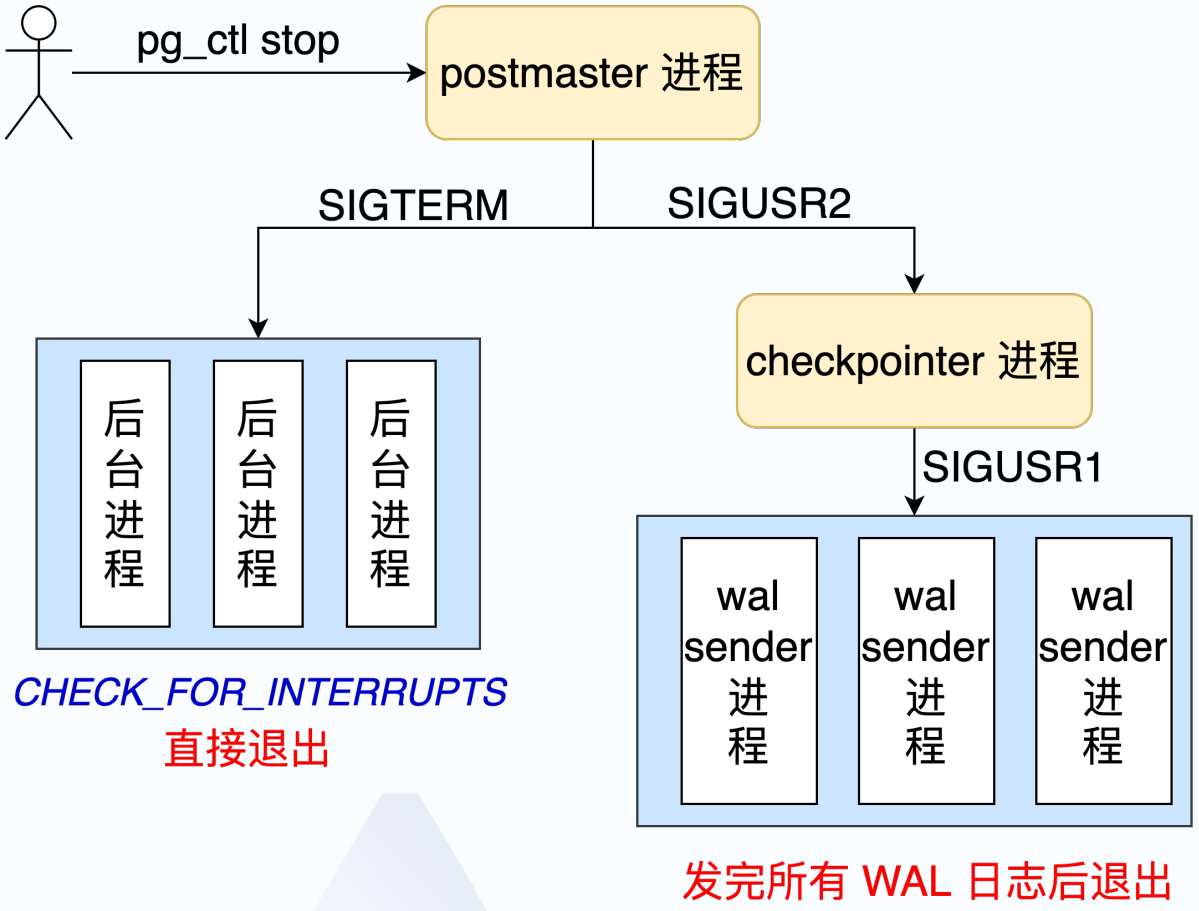
其余进程

```
#0  __epoll_wait_nocancel ()
#1  WaitEventSetWait ()
#2  WaitLatchOrSocket ()
#3  PgstatCollectorMain.isra.22 ()
#4  pgstat_start ()
#5  sigusr1_handler ()
#6  <signal handler called>
#7  __select_nocancel ()
#8  ServerLoop ()
#9  PostmasterMain ()
#10 main ()
```

逻辑复制实践经验分享

PG 实例重启卡住

- **原因**：正在运行的 walsender 进程会将所有 WAL 日志全部发送完毕再退出。



walsender 进程退出原理：

- replication 未开始：给自己发 SIGTERM 信号
- replication 已开始：CHECK_FOR_INTERRUPTS 不处理退出逻辑，等待实例现有的所有 WAL 日志发送完毕后退出。

PG 实例重启卡住

- **优化**：改进重启流程

  1. pg_ctl stop –m fast
  2. 若超过设定的时间阈值，改用 pg_ctl stop –m immedate 模式

- **为什么可以这么优化**：

  1. pg_ctl stop 的不同模式
     - smart 模式：不接受新的连接，等待现有的连接退出；
     - fast 模式（默认）：发送 SIGTERM 信号，子进程通过 CHECK_FOR_INTERRUPTS 捕获后退出，存在 walsender、checkpointer 等特例；
     - immediate 模式：发送 SIGQUIT 信号，所有子进程收到后立即退出，再次启动后进入 crash recovery 状态；
  2. 逻辑复制支持“断点续传”；

逻辑复制实践经验分享

逻辑复制 walsender 进程卡住

- **背景**：用户在使用逻辑复制时，发现 walsender 进程卡在同一个位点很久（超过 1h），始终不继续向前推进

- **现场**：perf 图显示 CPU 卡在 hash_seq_search 函数上，pg_waldump 对应的 lsn 是一条 drop publication 语句。

进程堆栈

```
#0  hash_seq_search ()
#1  rel_sync_cache_publication_cb () from pgoutput.so
#2  CallSyscacheCallbacks ()
#3  ReorderBufferCommit ()
#4  DecodeXactOp ()
#5  LogicalDecodingProcessRecord ()
#6  XLogSendLogical ()
#7  WalSndLoop ()
#8  exec_replication_command ()
#9  PostgresMain ()
#10 ServerLoop ()
#11 PostmasterMain ()
#12 main ()
```

WAL 信息

rmgr: Transaction len (rec/tot): 475877/475877, tx: 1511906902, lsn: 1F627/6ADC7DB8, prev 1F627/6ADC7D80, desc: COMMIT 2022-12-08 11:22:54.989016 CST; inval msgs:

catcache 45 catcache 44 catcache 47 catcache 46 catcache 47 catcache 46 catcache 47 catcache 46　后面省略很多个 catcache

逻辑复制 walsender 进程卡住

- **问题复现（版本 ≤ PG 13)**：

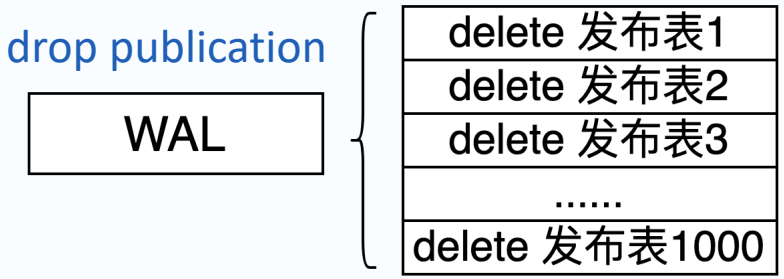
1. 创建两个 PG 实例，一个作为发布端，一个作为订阅端，创建一张表 t，并建立一个只包含表 t 的发布订阅；
2. 发布端创建 10w 张表；
3. 发布端创建一个发布 pub_large，只包含 1000 张表；
4. 发布端向 10w 张表每张表里插入一条数据；
5. 发布端运行 insert into t values(1);，此时订阅端能很快接收到此条数据；
6. 发布端运行 drop publication pub_large;
7. 发布端运行 insert into t values(1);，此时订阅端无法接收到此条数据；

逻辑复制实践经验分享

逻辑复制 walsender 进程卡住

- **原因：重复遍历哈希表**
  1. pgoutput invalid cache 回调函数多次重复遍历哈希表（无修改情况）；
  2. drop publication → 从系统表中删除 1000 条元组 → 生成 4000 条 invalid cache 消息

     walsender 进程解析到该条 WAL 日志时，针对每条 invalid cache 消息都会从哈希表中删除对应元素；



RelfilenodeMap 哈希表：通过 value 找 key，顺序遍历哈希表所有元素并删除对应的

时间复杂度： O (1000 * 4000 * 100000) ≈ O (10^{11})

command-id 数量　　invalid cache 消息数量　　哈希表元素个数

逻辑复制实践经验分享

逻辑复制 walsender 进程卡住

- **TencentDB for PG 优化（版本 ≤ PG 13）**
  1. pgoutput 增加 lazy flag，避免无效重复遍历哈希表；
  2. 建立反向哈希表（relid → relfilenode）

     时间复杂度：O (1000 * 4000 * 1) ≈ O(10^6)

- **社区优化（版本 ≥ PG 14）**
  1. 新增一种 XLOG_XACT_INVALIDATIONS 日志类型，记录 invalid cache 消息

     时间复杂度：O (1000 * 4 * 100000) ≈ O(10^8)

03 优化与改进

优化与改进

TencentDB for PG 针对逻辑复制做了什么

- DTS (Data Transmission Service) **数据传输服务平台**
  - 数据迁移
  - 数据同步

- **Failover Slot**
  支持将 logical replication slot 从主库同步到备库。如果出现 HA ， 用户可以无感知地继续使用 failover slot，无需重新创建 slot 再走一遍全量+增量复制流程。

- **支持丰富的逻辑解码插件**
  - tencent_decoding
  - decoder_raw
  - wal2json
  - ……

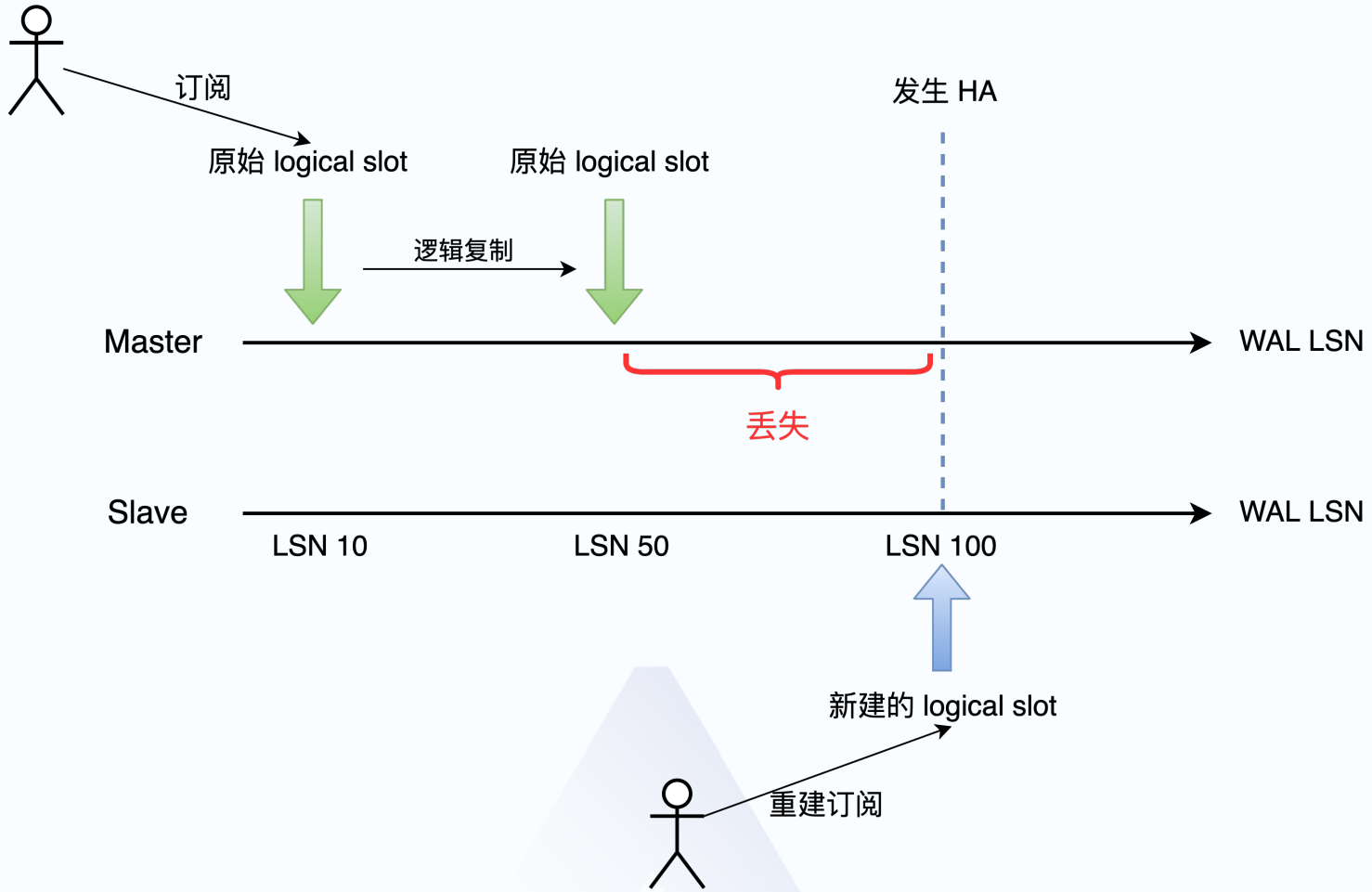
- **性能优化、bug 修复**

Failover Slot

- **问题背景：**

1. 当前数据库 LSN 100；
2. 逻辑复制只同步到 LSN 50；
3. 发生 HA 主备切换，slot 丢失；

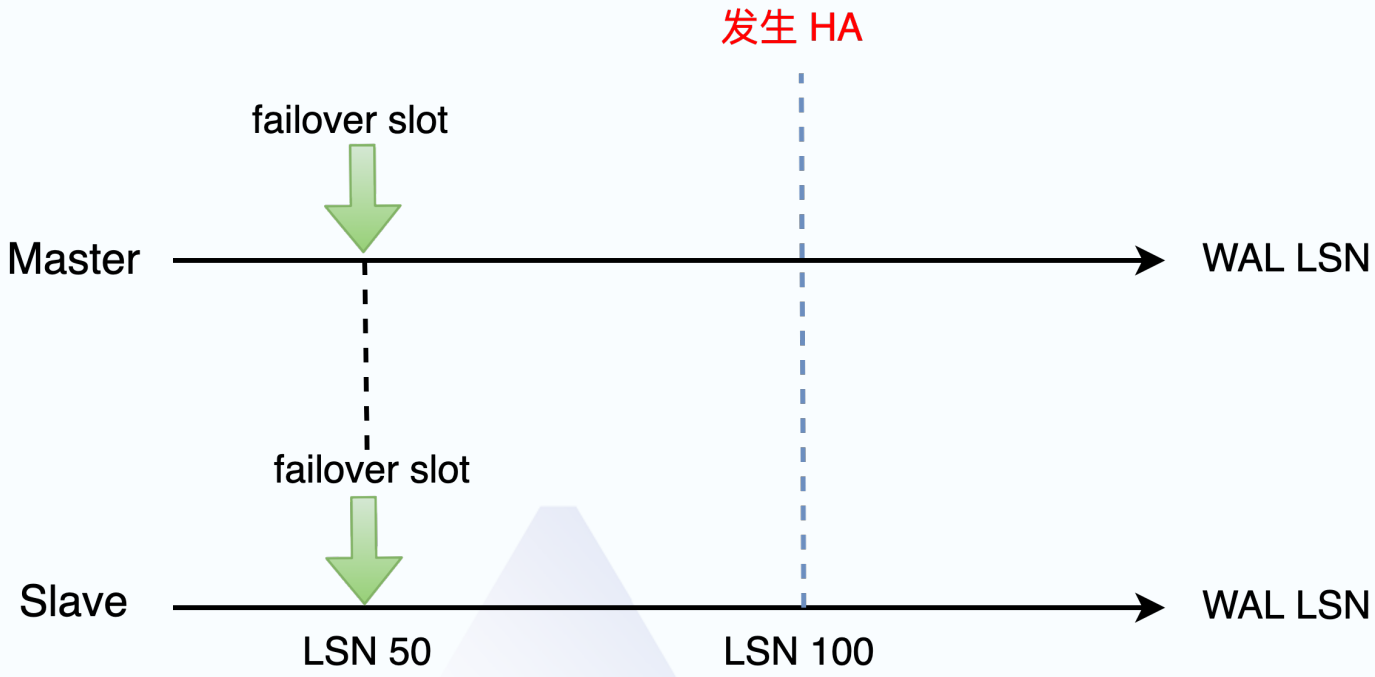
此时，由于缺少 LSN 50～100 的数据，需要用户重建订阅，走 **全量+增量** 流程。



Failover Slot

HA 后用户可无感知地继续使用原有的订阅

- **功能介绍：**
  - tencentdb_failover_slot 插件
    (创建/删除/查询 failover slot)
  - WAL 日志记录 slot 信息，主备同步；
  - 与社区 PG 同一大版本兼容
  - 针对异常情况做特殊处理

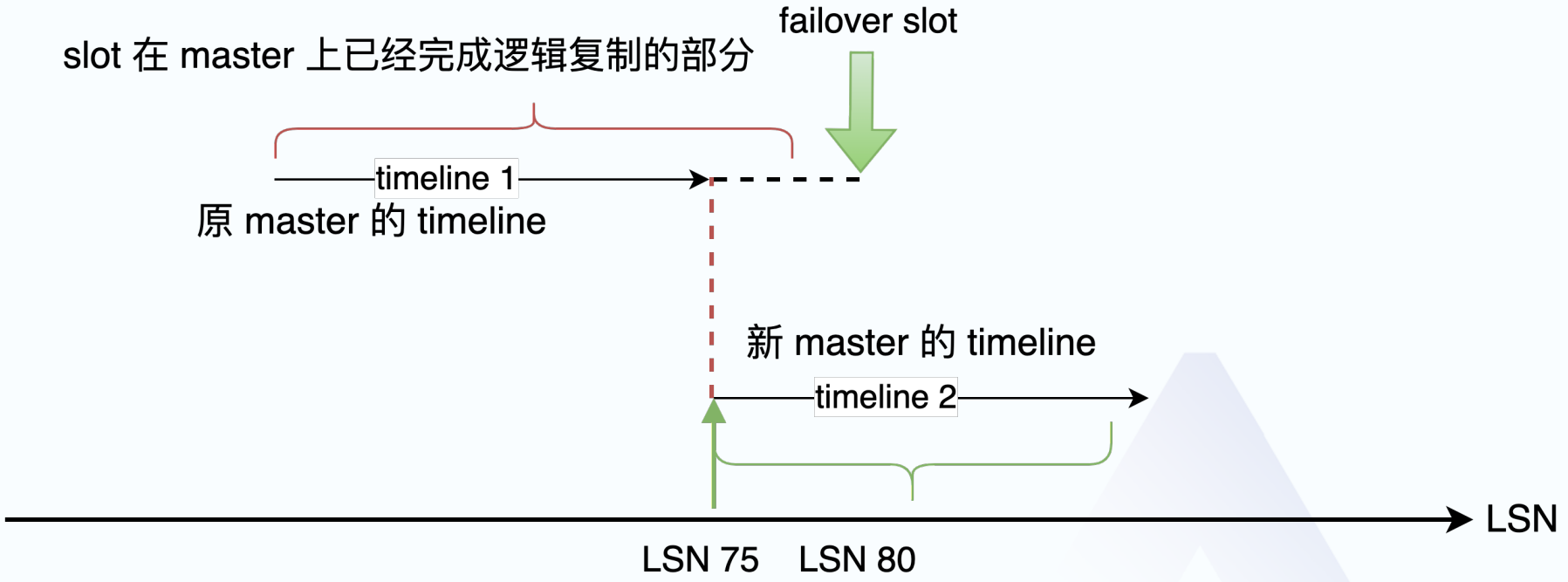


Failover Slot

- **异常场景：**逻辑复制比物理复制快

  提示出现异常，开放 GUC 供用户选择：1. 重建订阅；

  2. slot 回溯到 LSN 75 的位置继续订阅；



04 未来与展望

未来与展望

TencentDB for PG 未来

➢ 云原生化

- 打造基于云原生的操作系统
- 一云多芯

➢ 智能化

- AI4DB：数据库自治，协助 DBA 简化运维；
- DB4AI：在数据库中使用机器学习的算法处理、分析数据；

➢ Serveless 化

极致弹性，从卖资源转向卖能力，释放云计算资源池化潜力，实现客户与云厂商的双赢。

感谢聆听！

总想玩世不恭

江苏 南通

扫一扫上面的二维码图案，加我为朋友。