PostgreSQL中文社区

第13届PostgreSQL中国技术大会

聚焦云端创新　　汇聚智慧共享

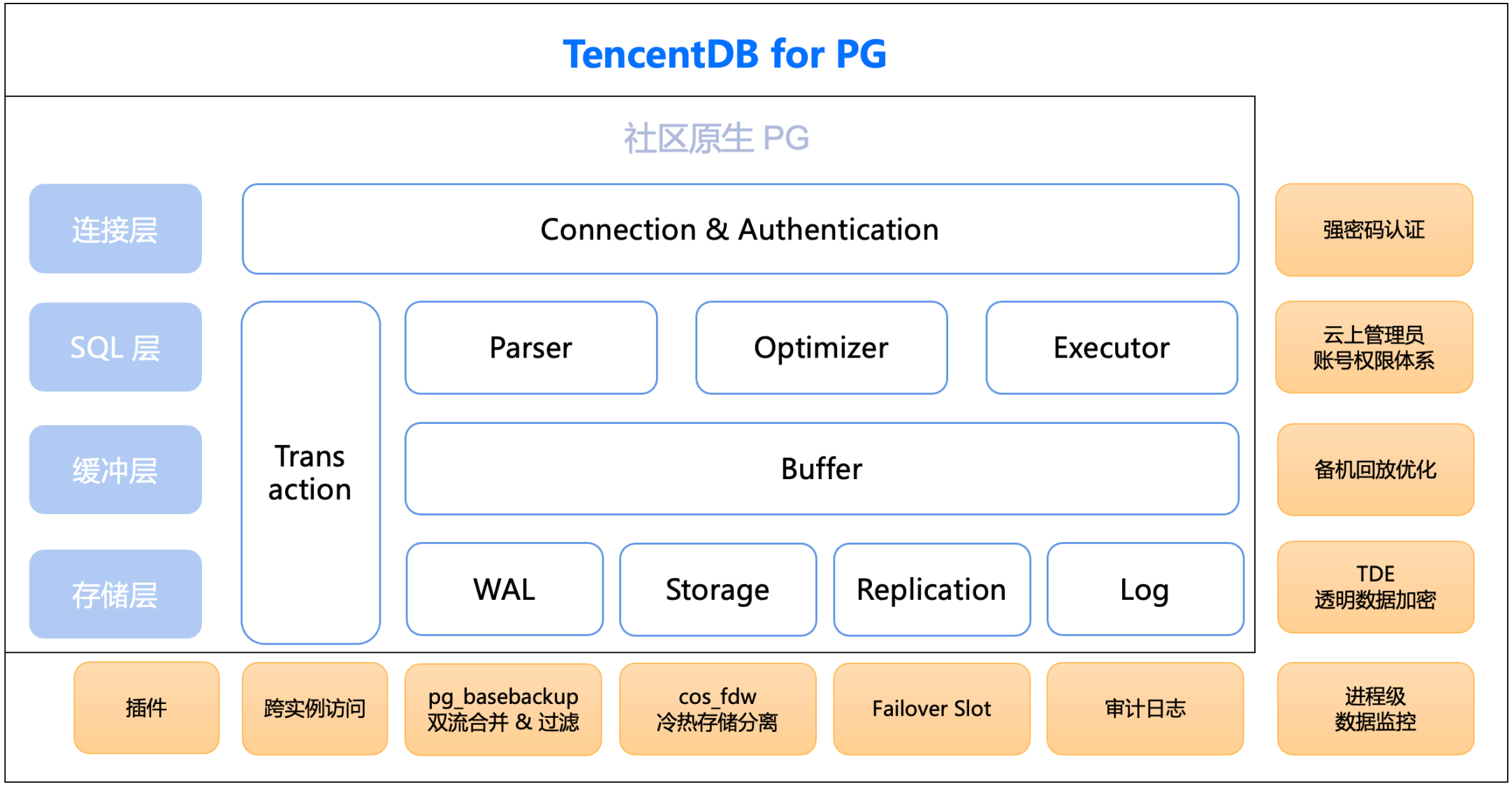
TencentDB for PG
内核原理揭秘

施博文

目录 contents

TencentDB for PG 内核功能全景图

PostgreSQL中文社区



01

进程级监控

01 进程级监控

➢ 功能介绍

- 统计指定时间内各进程 CPU 的使用率（相对于单核）
- 统计指定时间内各进程 内存 的使用量（单位 Byte）
- 通过 GUC 参数来设置每次采样的时长 （tencentdb_system_stat.sampling_interval)
- 提供 tencentdb_process_system_usage 视图供用户查询，可按照 database/进程/用户 分类，适用于 SaaS 场景

```
postgres=# select pid,query,cpu_usage,memory_bytes from tencentdb_process_system_usage ;
  pid  |                                    query                                    | cpu_usage | memory_bytes
-------+-----------------------------------------------------------------------------+-----------+--------------
 30744 |                                                                             |         0 |      7921664
 30741 |                                                                             |         0 |      4300800
 30742 |                                                                             |         0 |      8736768
 30767 | select sleep_and_loop2(0.66);                                               |      0.66 |     15556608
 30740 |                                                                             |         0 |      4300800
 30764 | select pid,query,cpu_usage,memory_bytes from tencentdb_process_system_usage ; |         0 |     16990208
 30739 |                                                                             |         0 |      4300800
(7 rows)
```

01 进程级监控

➢ 原理解析

- /proc/pid/stat : 记录对应进程号的状态信息

1. pid：进程号
2. comm：进程的命令名
3. state：进程状态
4. ppid：进程组 id

.....

14. utime：进程启动至今在用户态运行的时间（以 clock tick 为单位）
15. stime：进程启动至今在内核态运行的时间（以 clock tick 为单位）

.....

22. starttime：当前进程启动的时间，值表示距离操作系统启动已经经过的 clock tick 数；
24. rss：进程占用的实际内存大小（以 page 为单位）

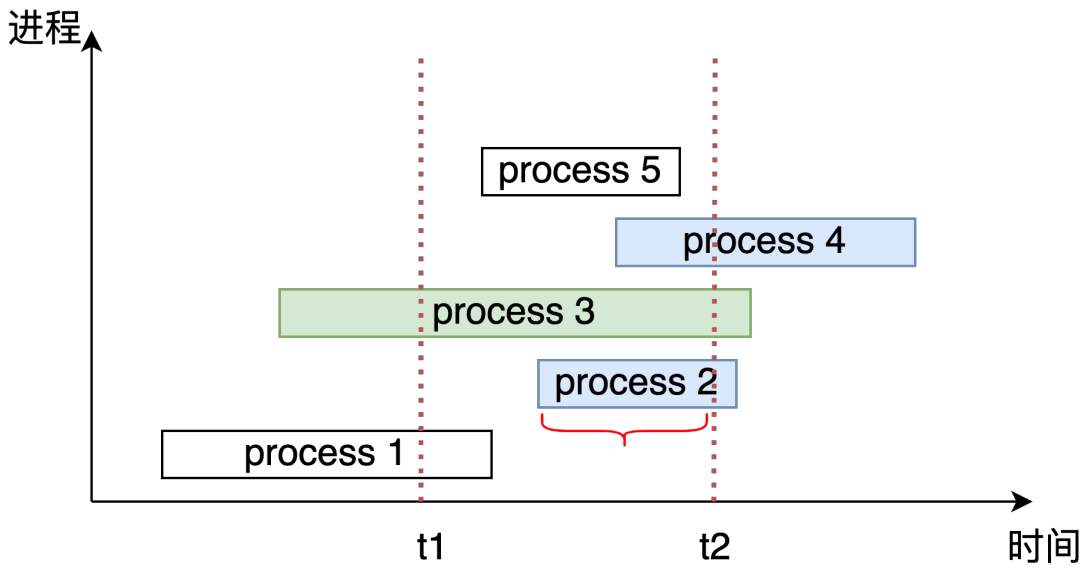
```
[postgr[redacted]]$ cat /proc/10130/stat
10130 (postgres) S 10128 10127 9464 0 -1 4194368 152 0 0 0 0 0 0 0 20 0 1 0 326881692 29393
3056 1598 18446744073709551615 4194304 15148540 140729905887232 0 0 0 4194304 24145920 5368
73479 1 0 0 17 16 0 0 0 0 0 17247552 17319640 33173504 140729905895460 140729905895502 1407
29905895502 140729905897433 0
```

01 进程级监控

➢ 原理解析

- 内存使用量计算：直接读取 rss 值
- CPU 使用率计算：需要计算 [t1,t2] 这段时间内，某个进程对 CPU 的占用情况

  jiffies：Linux 内核中的变量，记录从系统启动到现在已经经过的 clock tick 数 （记录在 /proc/uptime）



$$CPU使用率 = \frac{utime + stime}{jiffies_{now} - jiffies_{starttime}}$$

$$CPU使用率 = \frac{(utime_{t2} + stime_{t2}) - (utime_{t1} + stime_{t1})}{jiffies_{t2} - jiffies_{t1}}$$

- 绿色进程：通过公式 2 计算 CPU 使用率
- 蓝色进程：通过公式 1 计算 CPU 使用率
- 白色进程：不统计在内

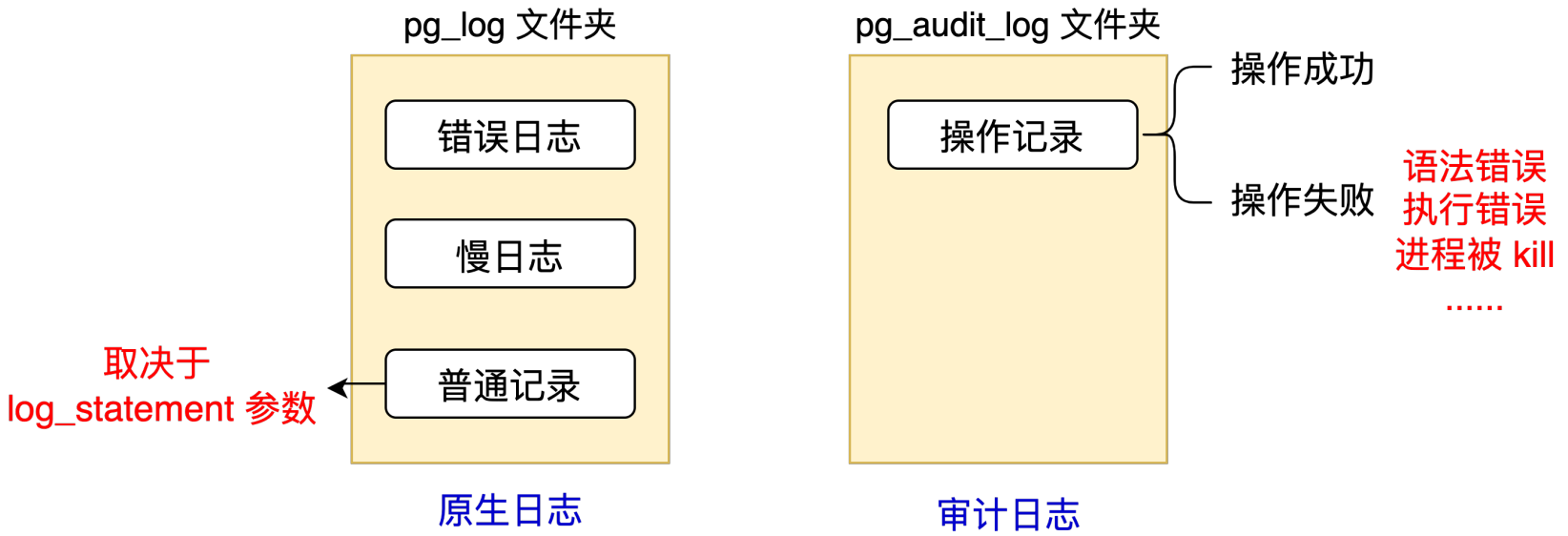
02

内核审计日志

02 内核审计日志

➢ 功能介绍

- 审计日志功能：记录数据库操作人员、操作对象、操作对象等一系列信息的日志。支持以下两种模式审计：
  1. 极速版：对标原生 log_statement=all ，新增审计影响行数、执行时长
  2. 精细版：对标 pgaudit 插件提供的能力，新增审计影响行数、执行时长
- 自定义 csv 格式审计日志文件，高可扩展性（对接 dbbrain）
- 审计日志与原生日志分流，保障高可读性和高性能

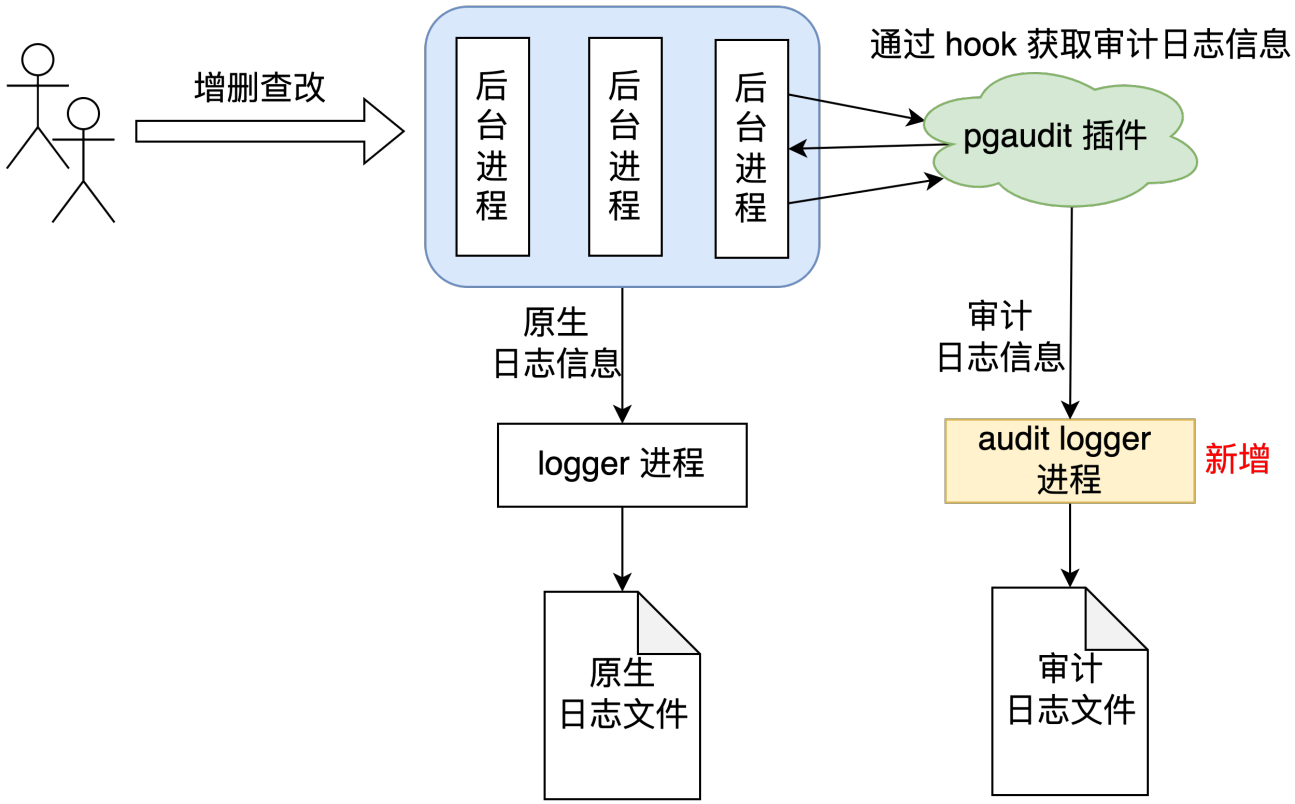


02 内核审计日志

➢ 原理解析

- 新增 audit logger 进程，后台进程通过 管道/ socket 将日志消息发送给该进程，该进程不断的将审计日志专门刷到对应的审计日志文件中：

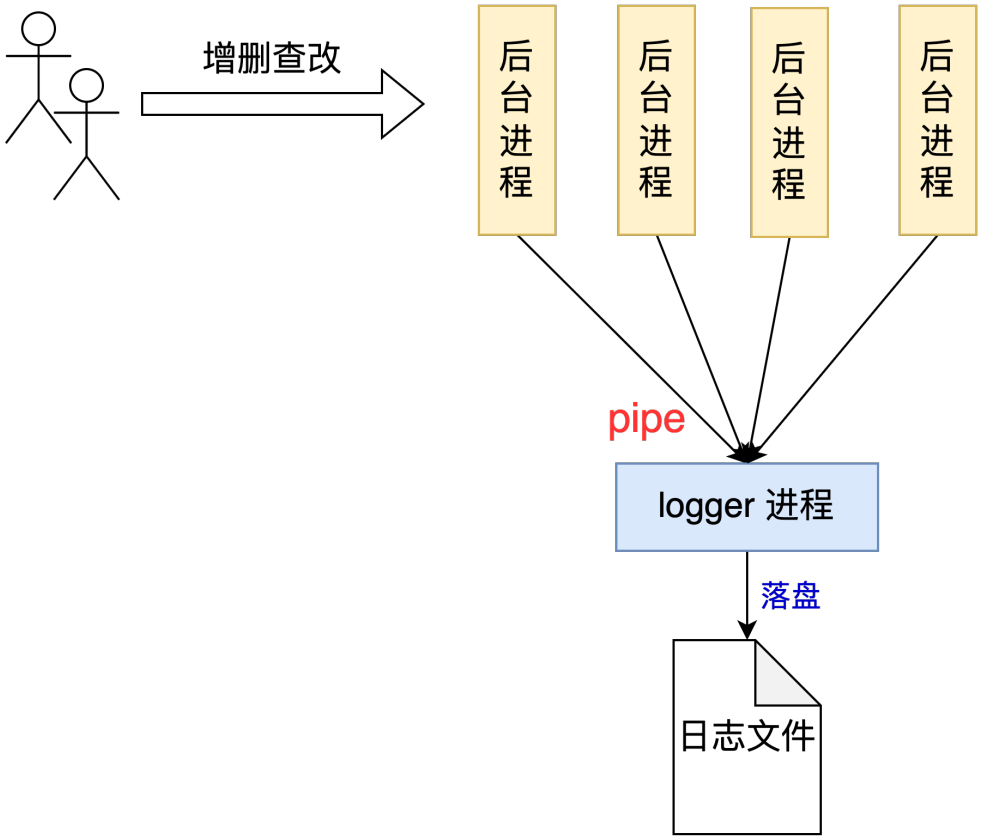
- 以 pgaudit 插件为基础进行修改，后台进程通过 hook 机制获取审计日志需要的信息，最终将其发送给 audit logger 进程



02 内核审计日志

➢ 性能优化

- PG 原生开启 log_statement=all 后性能下降明显，开启 pgaudit 后会进一步导致性能下降；
- 对性能瓶颈进行针对性优化：后台进程给 logger 进程发消息时会抢操作系统的 pipe 锁



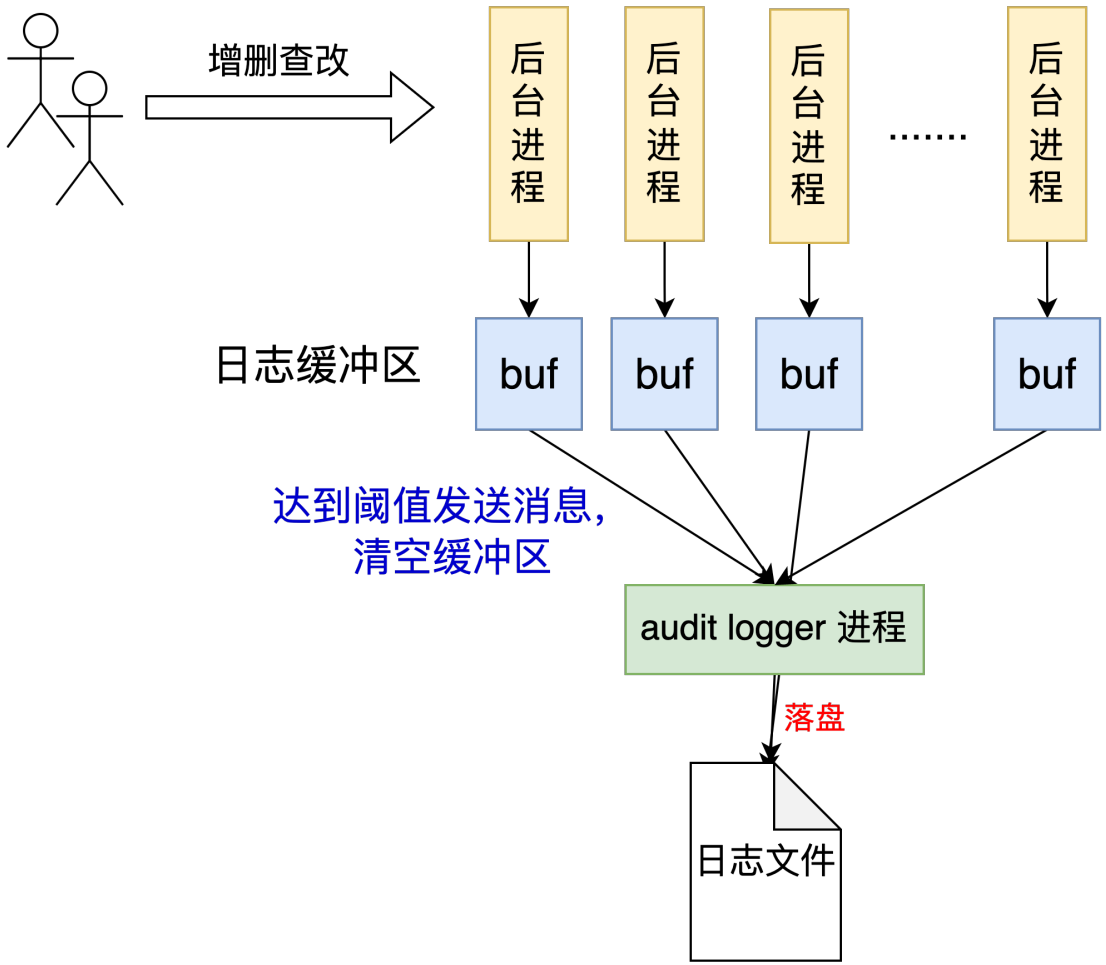

```
Samples: 1K of event 'cycles:ppp', Event count (approx.): 649534226
Overhead  Shared Object      Symbol
  71.56%  [kernel]           [k] osq_lock
   4.36%  [kernel]           [k] native_queued_spin_lock_slowpath
   1.29%  [kernel]           [k] find_busiest_group
   1.21%  [kernel]           [k] update_blocked_averages
   0.65%  [kernel]           [k] idle_bt_cpu
   0.61%  postgres           [.] hash_search_with_hash_value
   0.56%  libc-2.17.so       [.] __memcpy_ssse3_back
   0.54%  [kernel]           [k] mutex_spin_on_owner
   0.53%  postgres           [.] SearchCatCache1
   0.51%  [kernel]           [k] radix_tree_descend
   0.49%  [kernel]           [k] filemap_map_pages
   0.35%  [kernel]           [k] load_balance
   0.34%  [kernel]           [k] __list_del_entry_valid
   0.29%  [kernel]           [k] try_to_wake_up
   0.28%  libc-2.17.so       [.] __strncpy_sse2_unaligned
   0.28%  [kernel]           [k] update_curr
   0.26%  postgres           [.] namestrcpy
   0.21%  postgres           [.] LWLockRelease
   0.19%  [kernel]           [k] update_load_avg
   0.19%  postgres           [.] ExecResetTupleTable
   0.18%  postgres           [.] LWLockAcquire
   0.18%  [vdso]             [.] __vdso_gettimeofday
   0.17%  postgres           [.] TupleDescInitEntry
   0.17%  [kernel]           [k] __handle_mm_fault
   0.17%  [kernel]           [k] ep_poll
   0.17%  [ip_tables]        [k] ipt_do_table
   0.17%  postgres           [.] GetDatabaseEncoding
   0.17%  postgres           [.] PostgresMain
```

02 内核审计日志

➢ 性能优化

核心优化思路：减少 pipe 锁的争抢 → 减少后台进程发消息的频率 → 多攒几条日志消息一起发

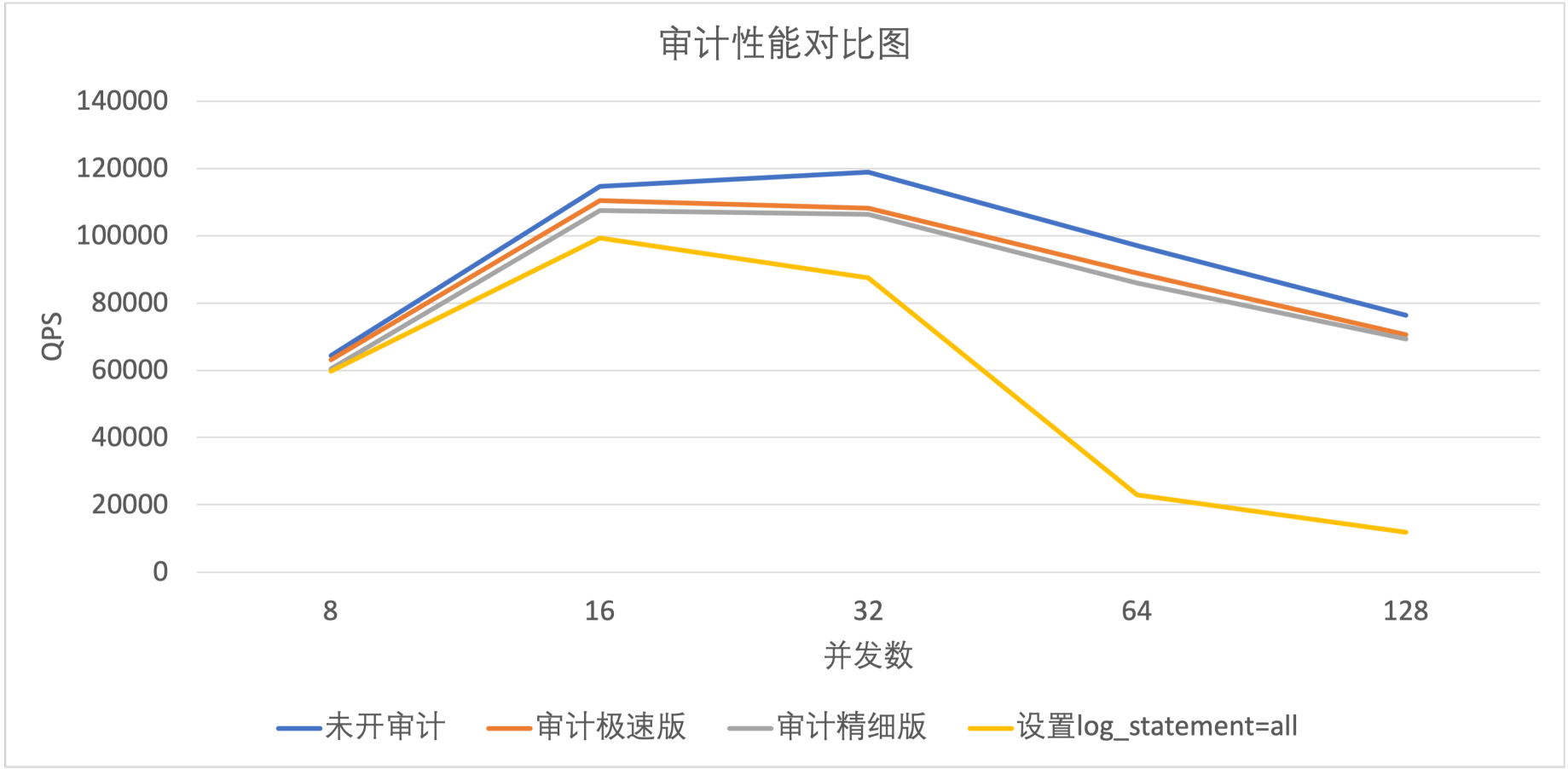
- 给每个后台进程建立自己的日志缓冲区
- 缓冲区大小可通过参数动态调整
- 缓冲区满将消息统一发送给 logger 进程
- 缓冲区内存复用，减少实例的内存使用量
- 将审计日志的落盘机制由 行缓冲 改为 全缓冲



02 内核审计日志

➢ 性能优化

现网 8C32G 实例的性能测试结果如下，高并发场景下性能提升明显



03

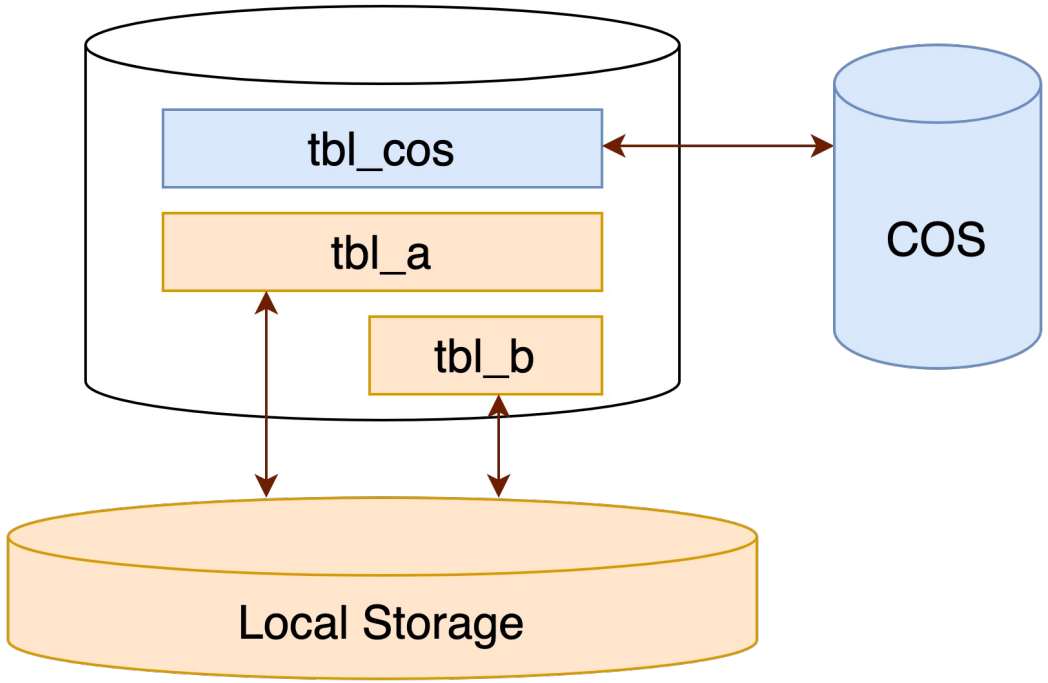
冷热存储分离

03 冷热存储分离

➢ 功能介绍

- 提供 cos_fdw 插件，用户可以将不常用的数据放到 COS（对象存储） 上，使用 cos_fdw 可直接将对象存储中的数据当作外部表来访问

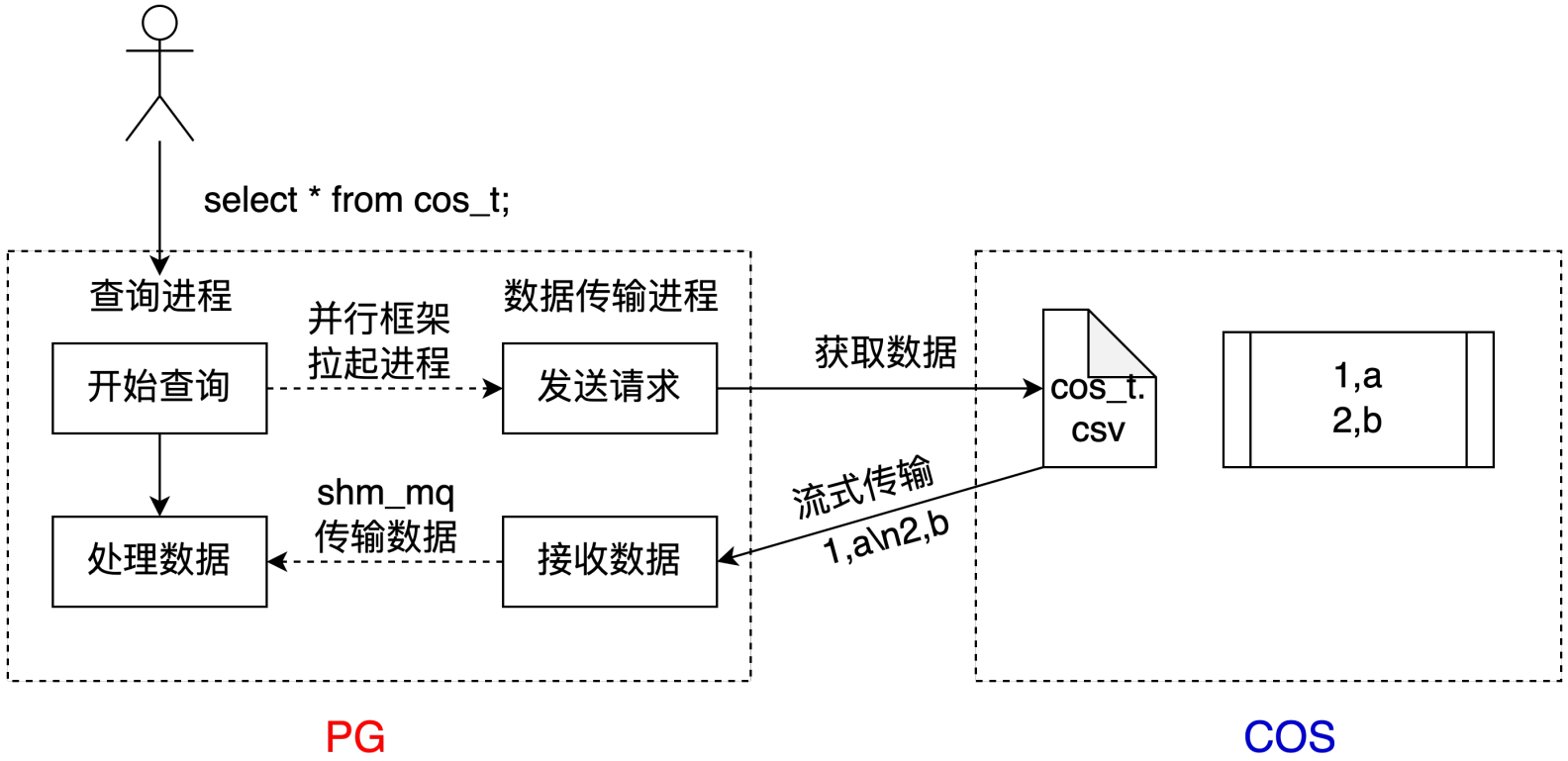
```
SELECT
        a.id,b.name,c.value
FROM
        tbl_a a,tbl_b b,tbl_cos cos
WHERE
        a.id = b.id ........
```



03 冷热存储分离

➢ 原理解析

- 利用 PG 原生的 FDW 机制，将 COS 上的 csv 文件当作外部表处理
- 通过网络与 COS 进行交互，流式传输数据
- 数据传输、处理并行优化
- COS 的用户信息加密存储



04

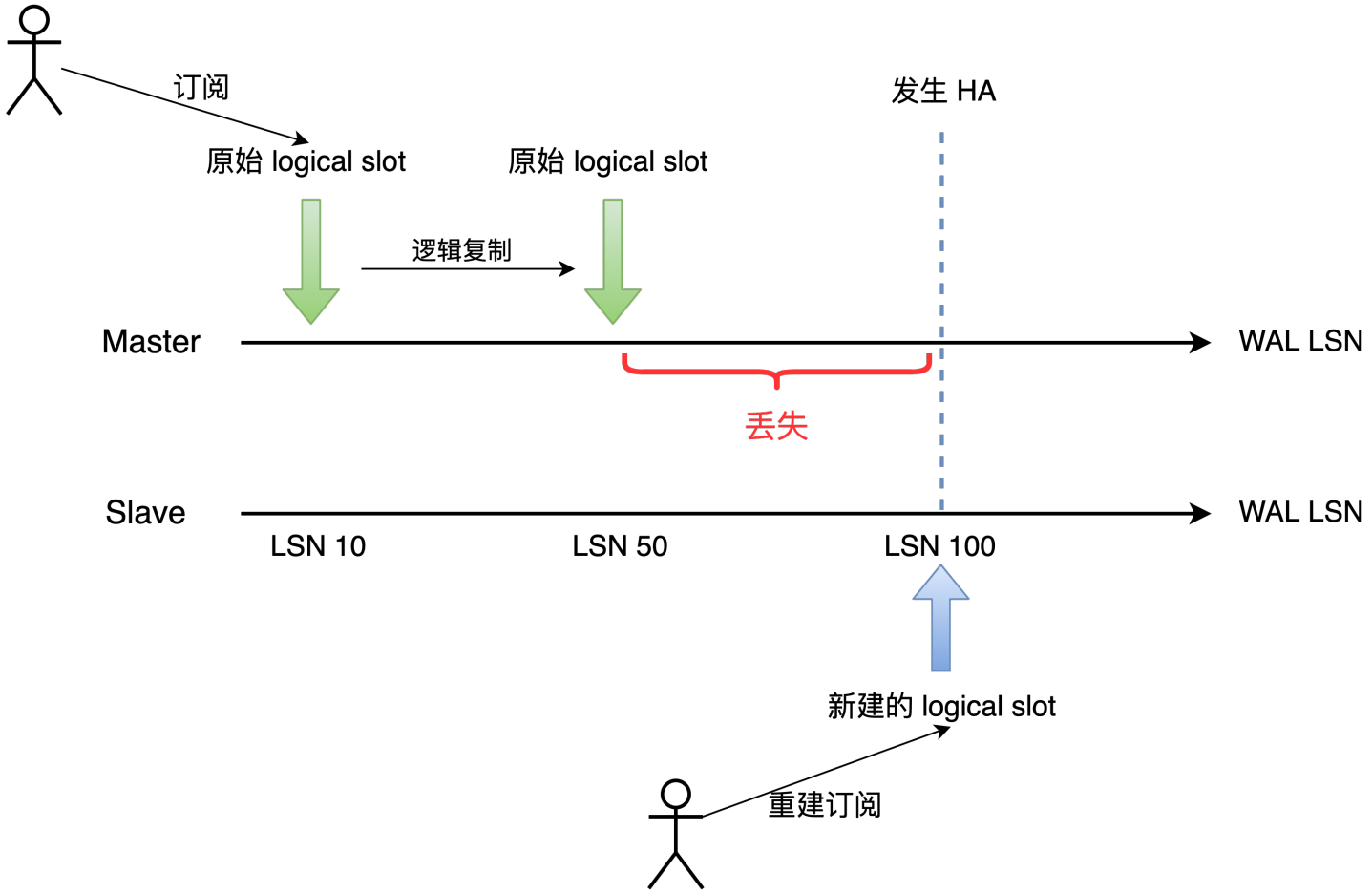
逻辑复制槽故障转移

04 逻辑复制槽故障转移

➢ 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。



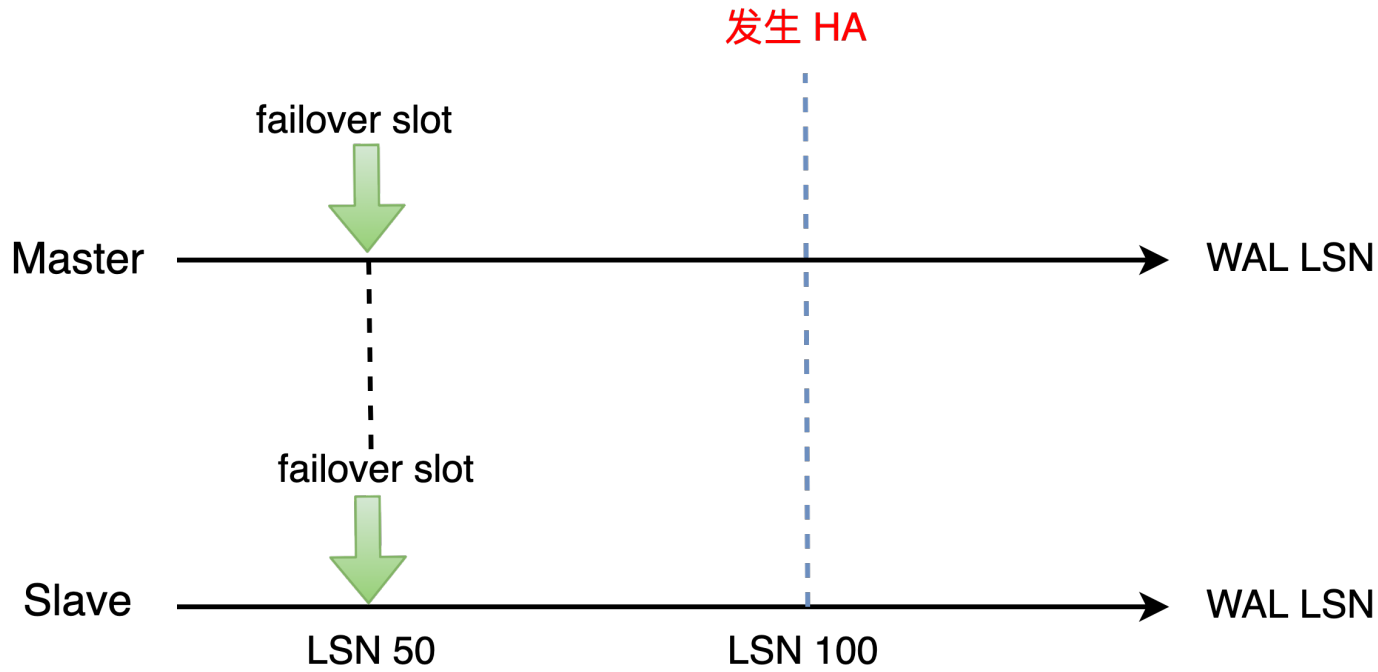
04 逻辑复制槽故障转移

➢ 功能介绍

- 逻辑复制槽自动从主库同步到备库
- HA 后用户可无感知地继续使用原有的订阅

➢ 原理解析

- 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

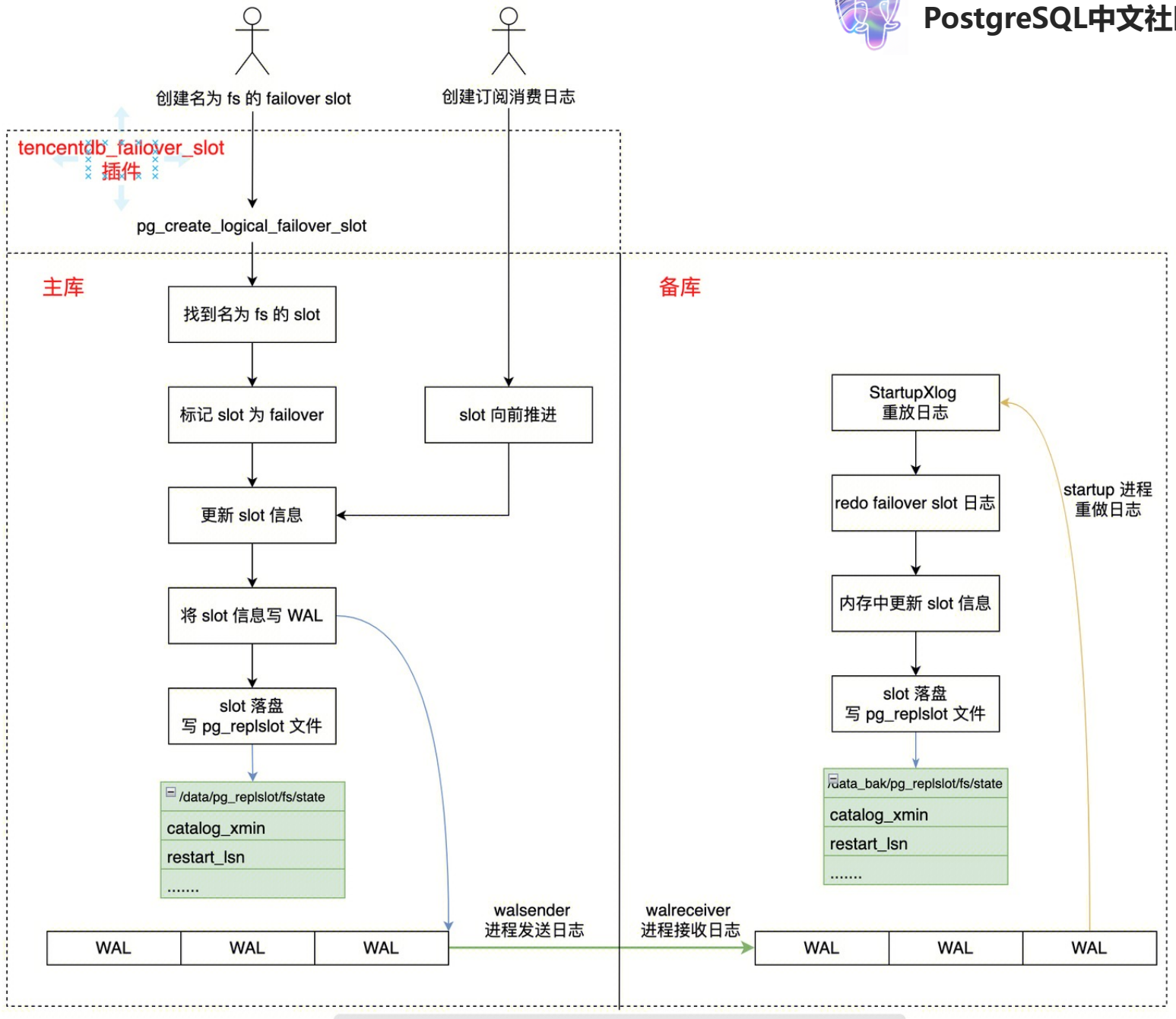


04 逻辑复制槽故障转移

PostgreSQL中文社区

➢ 原理解析

- 主库创建/更新 slot
- slot 的更新动作记录到 WAL
- 备库 redo 对应类型的 WAL



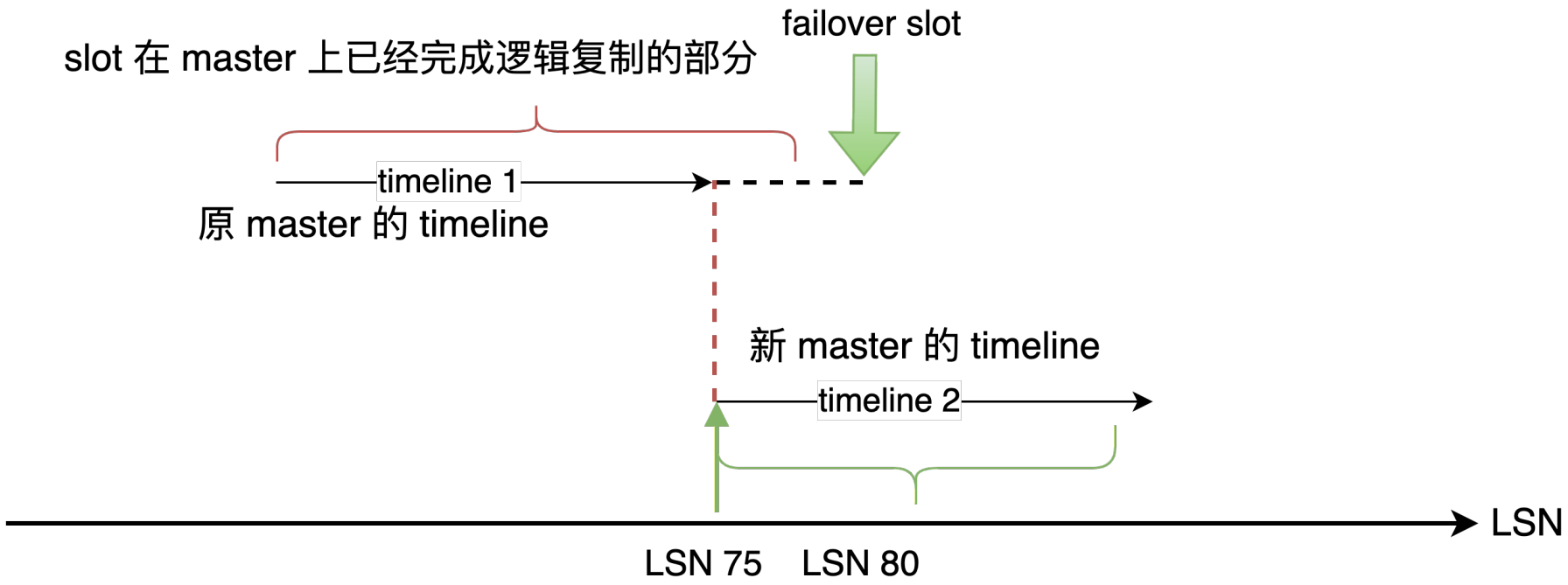
04 逻辑复制槽故障转移

➢ 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；

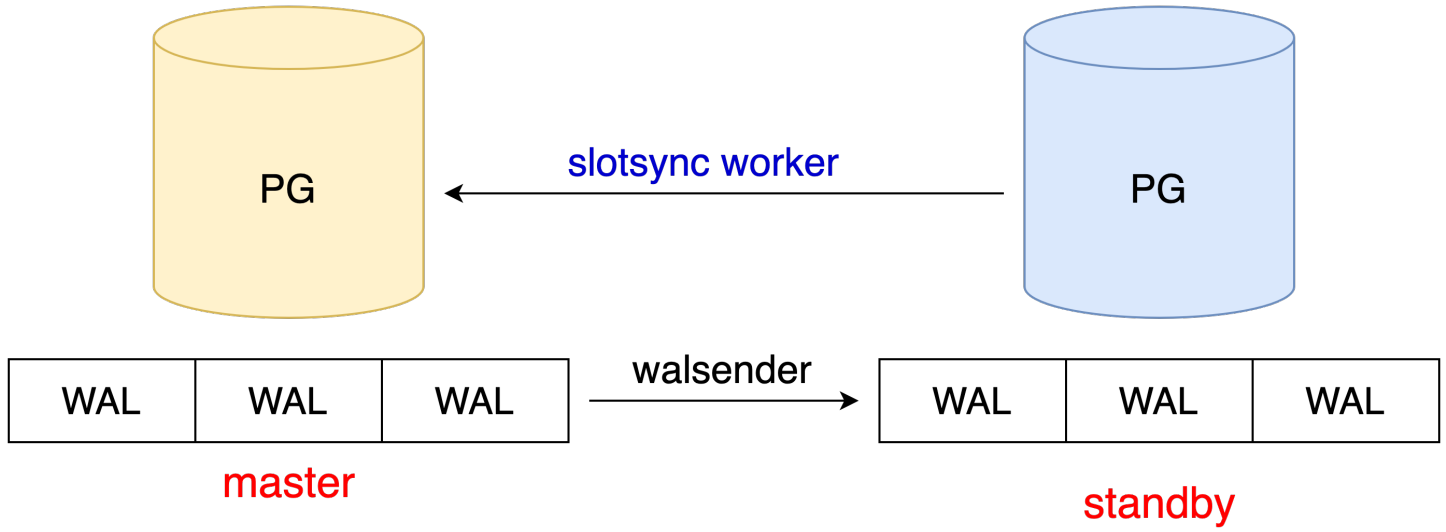
  2. slot 回溯到 LSN 75 的位置继续订阅；



04 逻辑复制槽故障转移

➢ PG 17 failover slot

- 备库通过 slotsync woker 经常定期轮询主库同步 slot 元数据
- *synchronized_standby_slots* 参数控制逻辑复制 慢于 指定的物理复制



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。