PostgreSQL/AntDB Replication Internal

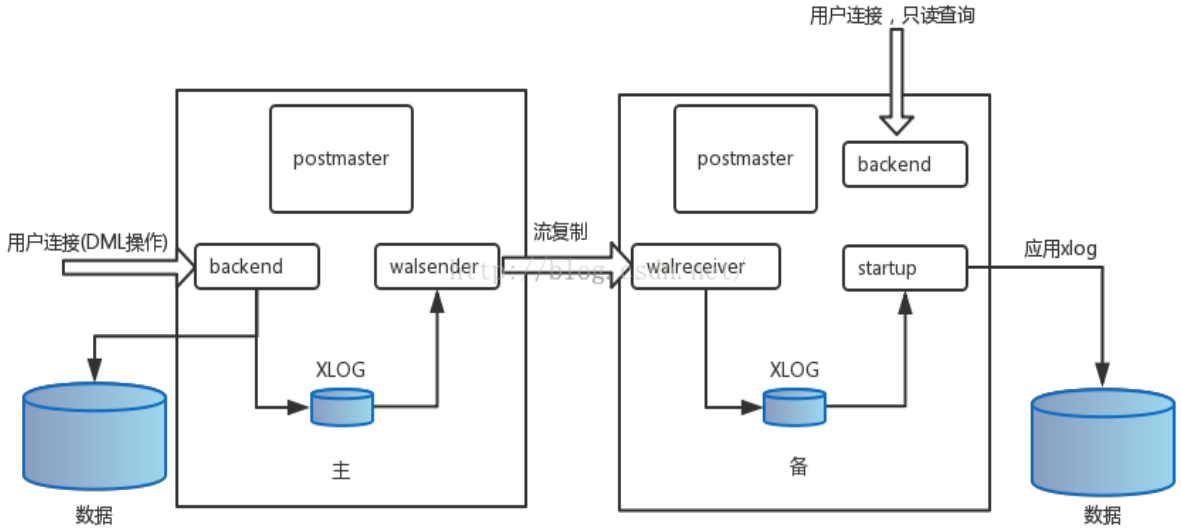
亚信 党宏博

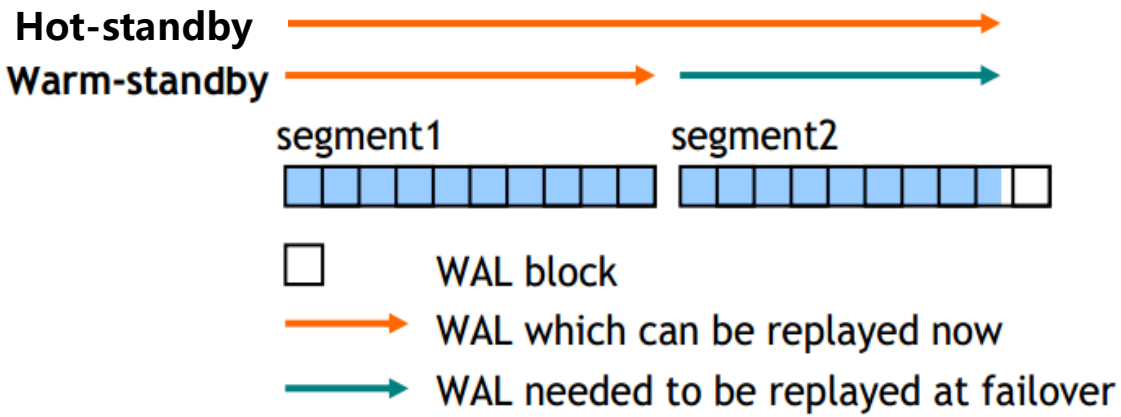
内容

- Replication-concept /流复制的基本概念
- Replication-Internal architecture/流复制的内部**实现**
- Replication-AntDB Extension Features/AntDB**扩**展功能
- Replication-Tools/基于流复制的管理工具

什么是流复制？

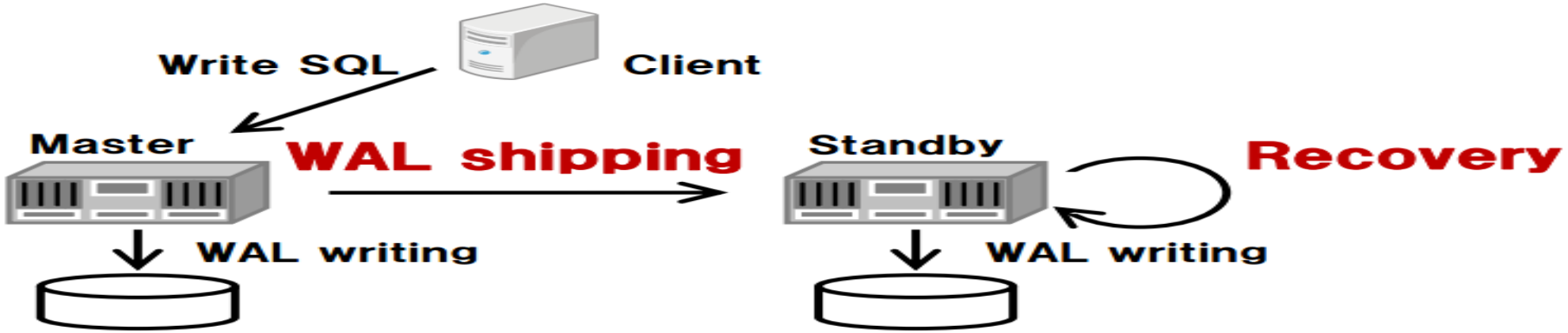
- 什么是复制：在多台主机之间建立数据副本的一种方式
- 什么是流复制：备机不断的接受master节点的WAL事务日志流并进行apply的过程
- 什么是逻辑复制：将物理变更转化成逻辑变更并应用到备节点的表对象。
- 区别：流复制针对的是实例级别，逻辑复制可以针对的是表级别





流复制的发展历程

从PG9.0开始引入了流复制功能



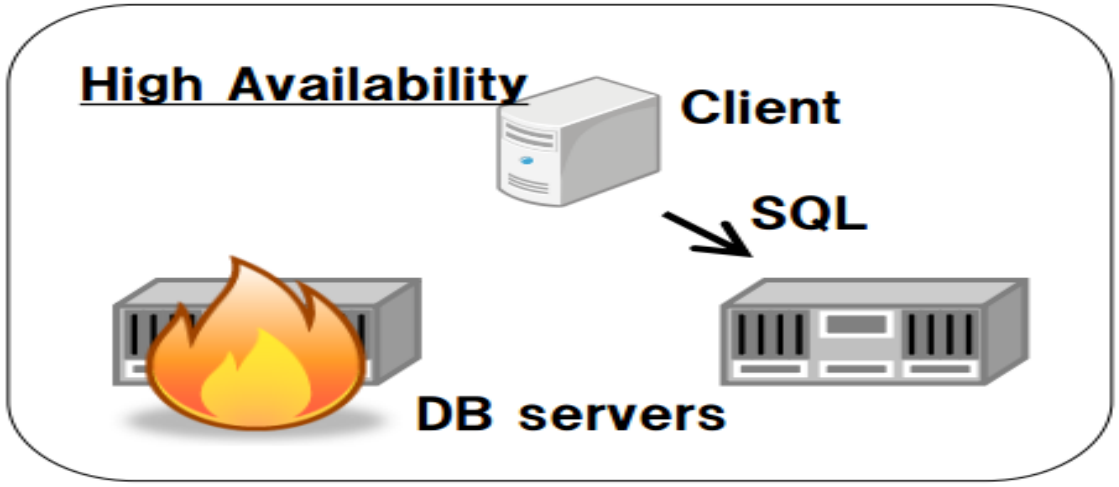
流复制的作用

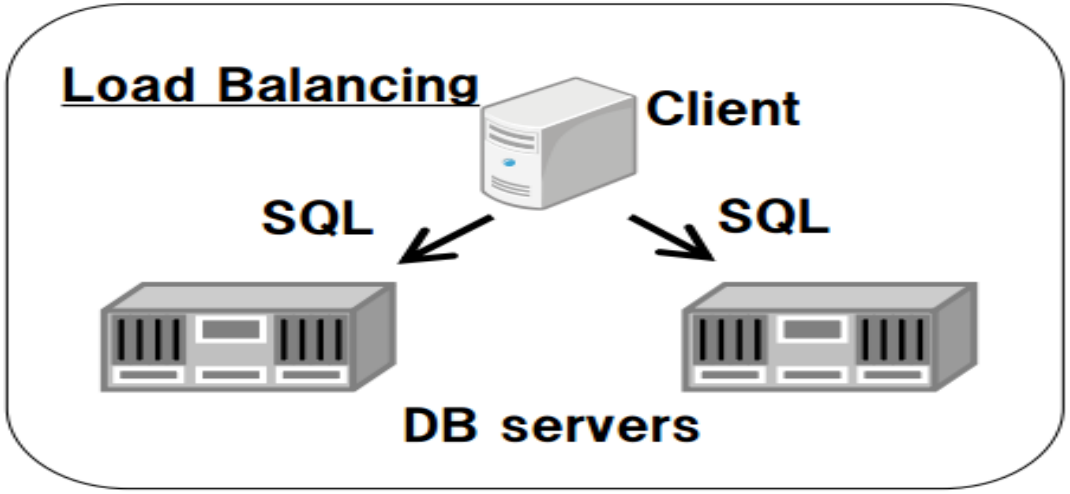
高可用

- 如果一台服务器崩溃，另一台服务器可以继续提供服务，可以轻松减少系统的停机时间

负载均衡

- 查询的负载可以分布到多个服务器执行实现读写分离，从而可以提高整个系统的性能





流复制相关基本概念-WAL

- WAL日志也称为事务日志或Redo日志
- 每一个事务日志都有一个唯一递增的LSN号，用64位进行存储
- 数据库发生的任何改变都会持久化在WAL日志文件中，一旦执行commit就会触发fsync将事务日志从内存中刷新到磁盘中

```
my_test=# set wal_debug to on
my_test-# ;
SET
my_test=#
my_test=#
my_test=# select count(*) from t1;
 count
-------
 10005
(1 row)

my_test=# delete from t1 where id=1;
LOG:  INSERT @ 0/63001E68: prev 0/63000090; xid 16721; len 26; bkpb0: Heap - delete: rel 1663/16384/24576; tid 0/1 KEYS_UPDATED
LOG:  INSERT @ 0/63003EB8: prev 0/63001E68; xid 16721; len 26: Heap - delete: rel 1663/16384/24576; tid 0/6 KEYS_UPDATED
LOG:  INSERT @ 0/63003EF8: prev 0/63003EB8; xid 16721; len 12: Transaction - commit: 2014-10-13 09:42:53.109066-07
LOG:  xlog flush request 0/63003F28; write 0/63001E68; flush 0/63001E68
DELETE 2
my_test=#
```

WAL log debug output

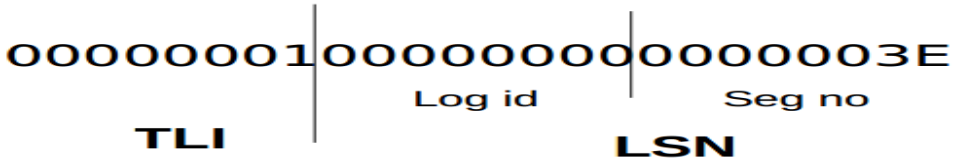
已连接 192.168.79.134:22。

流复制相关基本概念-WAL

事务日志存储在pg_wal目录下，每个wal日志文件为16M

```
[root@localhost pg_xlog]# ll
total 196616
-rw-------  1 postgres postgres      305 Oct 13 08:53 000000010000000000000061.00000028.backup
-rw-------. 1 postgres postgres 16777216 Oct 15 05:00 000000010000000000000078
-rw-------. 1 postgres postgres 16777216 Oct 15 06:01 000000010000000000000079
-rw-------. 1 postgres postgres 16777216 Oct 15 08:00 00000001000000000000007A
-rw-------  1 postgres postgres 16777216 Oct 15 09:00 00000001000000000000007B
-rw-------  1 postgres postgres 16777216 Oct 15 10:00 00000001000000000000007C
-rw-------  1 postgres postgres 16777216 Oct 15 11:00 00000001000000000000007D
-rw-------  1 postgres postgres 16777216 Oct 15 12:00 00000001000000000000007E
-rw-------. 1 postgres postgres 16777216 Oct 15 14:00 00000001000000000000007F
-rw-------. 1 postgres postgres 16777216 Oct 15 15:00 000000010000000000000080
-rw-------. 1 postgres postgres 16777216 Oct 15 16:00 000000010000000000000081
-rw-------. 1 postgres postgres 16777216 Oct 15 16:21 000000010000000000000082
-rw-------. 1 postgres postgres 16777216 Oct 15 03:00 000000010000000000000083
drwx------. 2 postgres postgres     4096 Oct 15 16:05 archive_status
[root@localhost pg_xlog]#
```

WAL 文件名格式



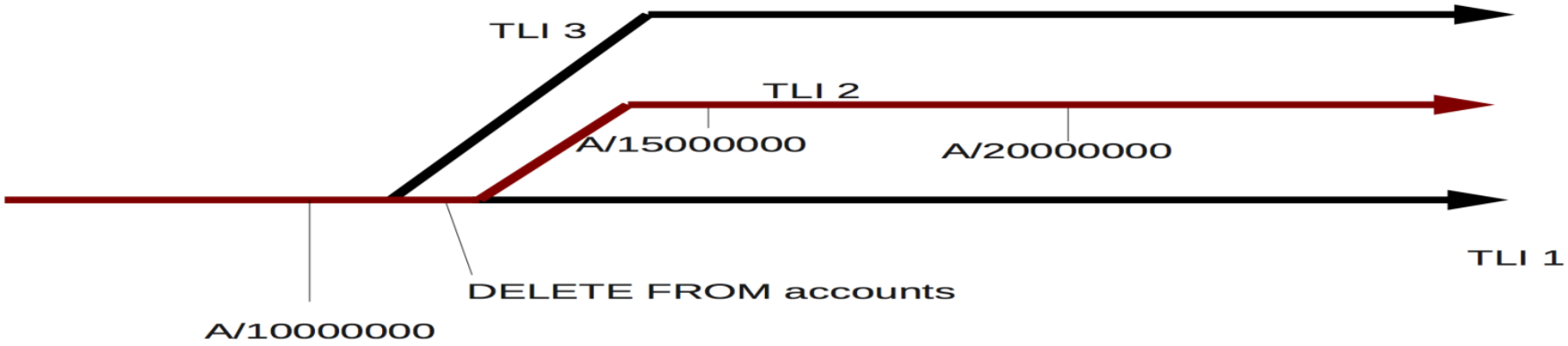
同一个时间线(TLI)里面, 当XLOG_SEG_SIZE=16MB时. 最多可以产生 2^32 * 256 个 XLOG 文件. 如果我们的数据库平均1天产生10TB的WAL LOG数据量, 那么需要

（（2^32 * 256）*16*1024*1024)

/(10*1024*1024*1024*1024)=1677721天=4721年

流复制相关基本概念-Timeline

使用PITR基于时间点的恢复操作后和备节点提升为主节点后都会改变时间线，从而避免对Old WAL 日志文件产生覆盖




```
000000010000000130000000E1
000000010000000130000000E2
000000010000000130000000E3
000000010000000130000000E4
000000010000000130000000E5
```

```
000000020000000130000000E3
000000020000000130000000E4
000000020000000130000000E5
```

流复制相关基本概念-Timeline history file

000000010000000130000000E1
000000010000000130000000E2
000000010000000130000000E3
000000010000000130000000E4
000000010000000130000000E5
00000002.history
000000020000000130000000E3
000000020000000130000000E4
000000020000000130000000E5

$ cat data-standby1/pg_xlog/00000003.history

```
1  13/E4000000  no recovery target specified
2  13/ED000090  at restore point "before FOSDEM"
```

Timeline ID

Point in WAL where new timeline begins

Reason for the timeline switch

流复制相关基本概念-LSN

```
src/include/access/xlogdefs.h
00017 /*
00018  * Pointer to a location in the XLOG.  These pointers are 64 bits wide,
00019  * because we don't want them ever to overflow.
00020  */
00021 typedef uint64 XLogRecPtr;
```

WAL的LSN寻址上限是 2^64 和 64位机器的内存寻址空间一样

可以通过pg_current_xlog_insert_location函数获取系统当前的LSN号

```
postgres=# select pg_current_xlog_insert_location();
 pg_current_xlog_insert_location
---------------------------------
 0/8ECABFB0
(1 row)

postgres=#
```

通过LSN号获取该记录所在的事务日志文件

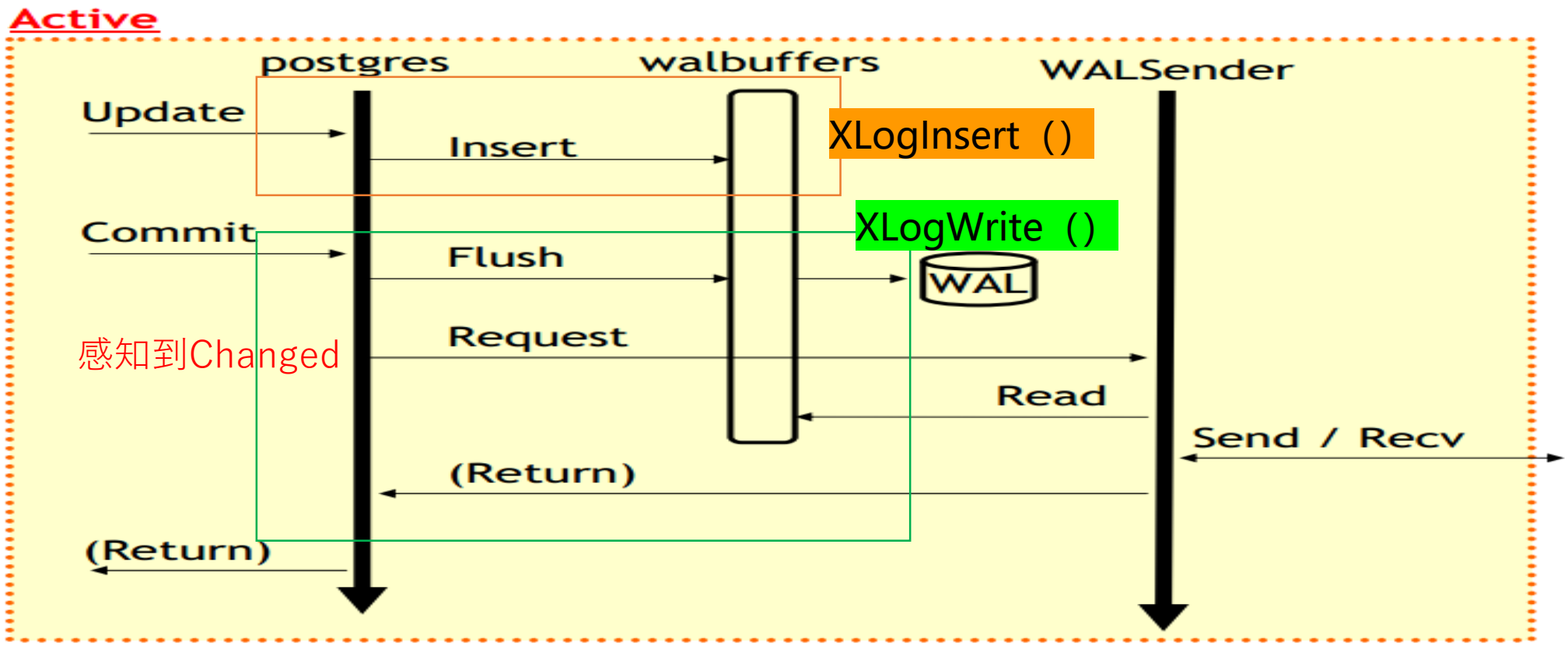
```
postgres=# select pg_xlogfile_name('0/667730c8');
DEBUG:  StartTransactionCommand
DEBUG:  StartTransaction
DEBUG:  name: unnamed; blockState:       DEFAULT; state: INPROGR, xid/subid/cid: 0/1/0, nestlvl: 1, children:
DEBUG:  CommitTransactionCommand
DEBUG:  CommitTransaction
DEBUG:  name: unnamed; blockState:       STARTED; state: INPROGR, xid/subid/cid: 0/1/0, nestlvl: 1, children:
     pg_xlogfile_name
--------------------------
 000000010000000000000066
(1 row)

postgres=#
```

内容

- Replication-concept /流复制的基本概念
- Replication-Internal architecture/流复制的内部**实现**
- Replication-Features/流复制功能介**绍**
- Replication-Configure file/流复制相关配置文件

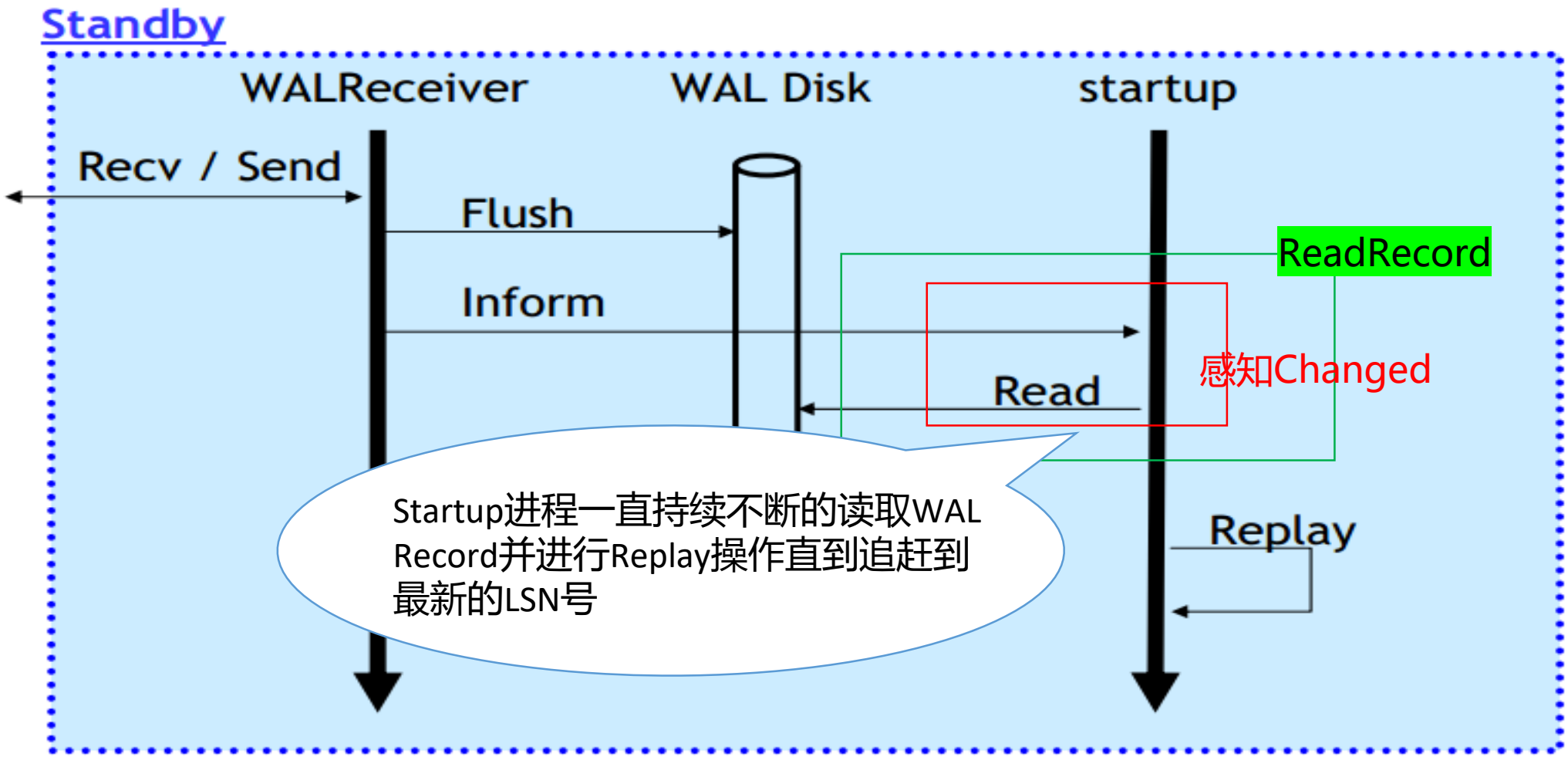
流复制-Internal architecture

Postgres 后台服务进程和WALSender进程间的触发关系



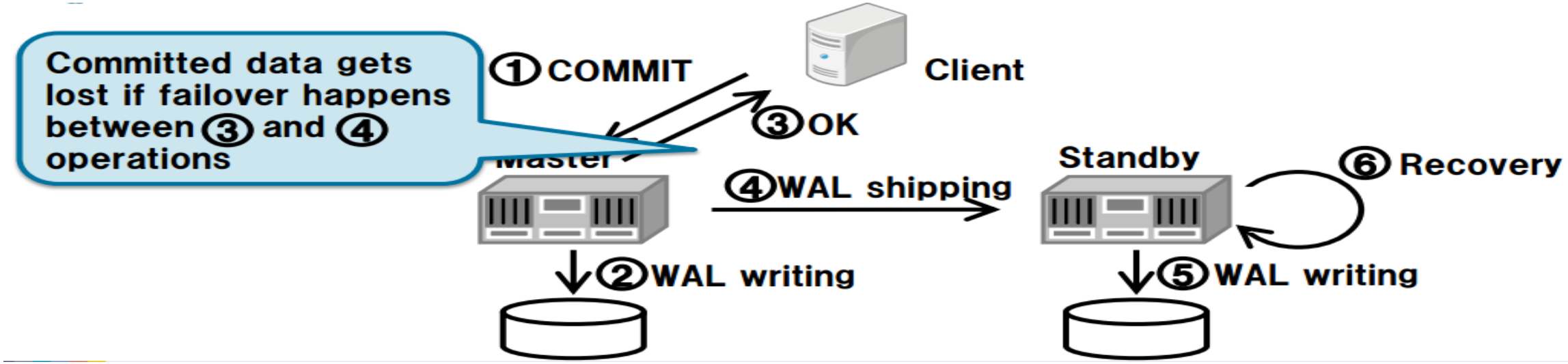
流复制-Internal architecture

WALReceiver进程和StartUP 进程间的触发机制



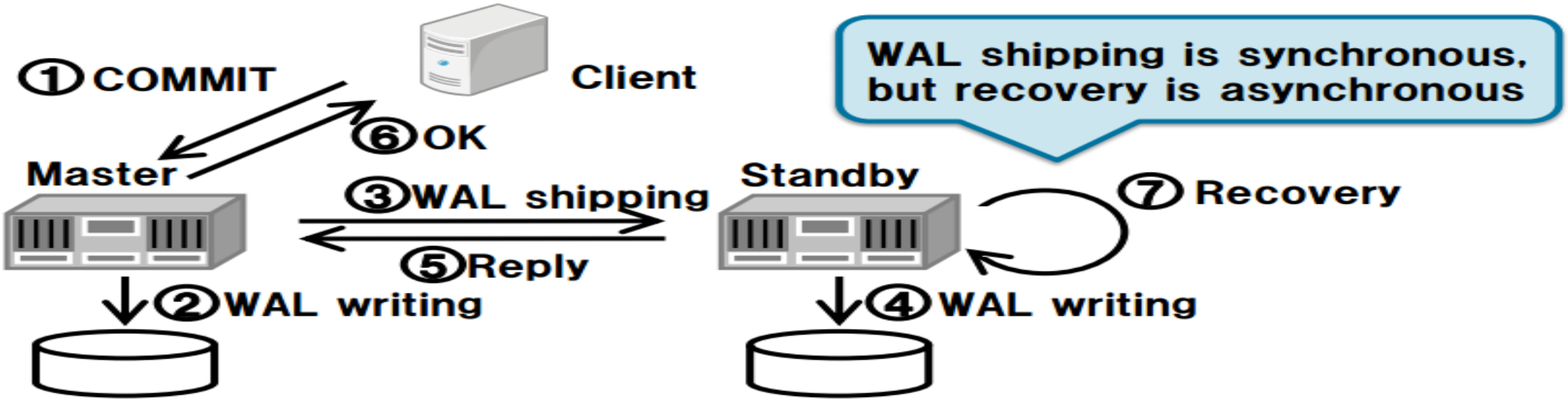
流复制-Internal architecture

异步复制的实现逻辑



流复制-Internal architecture

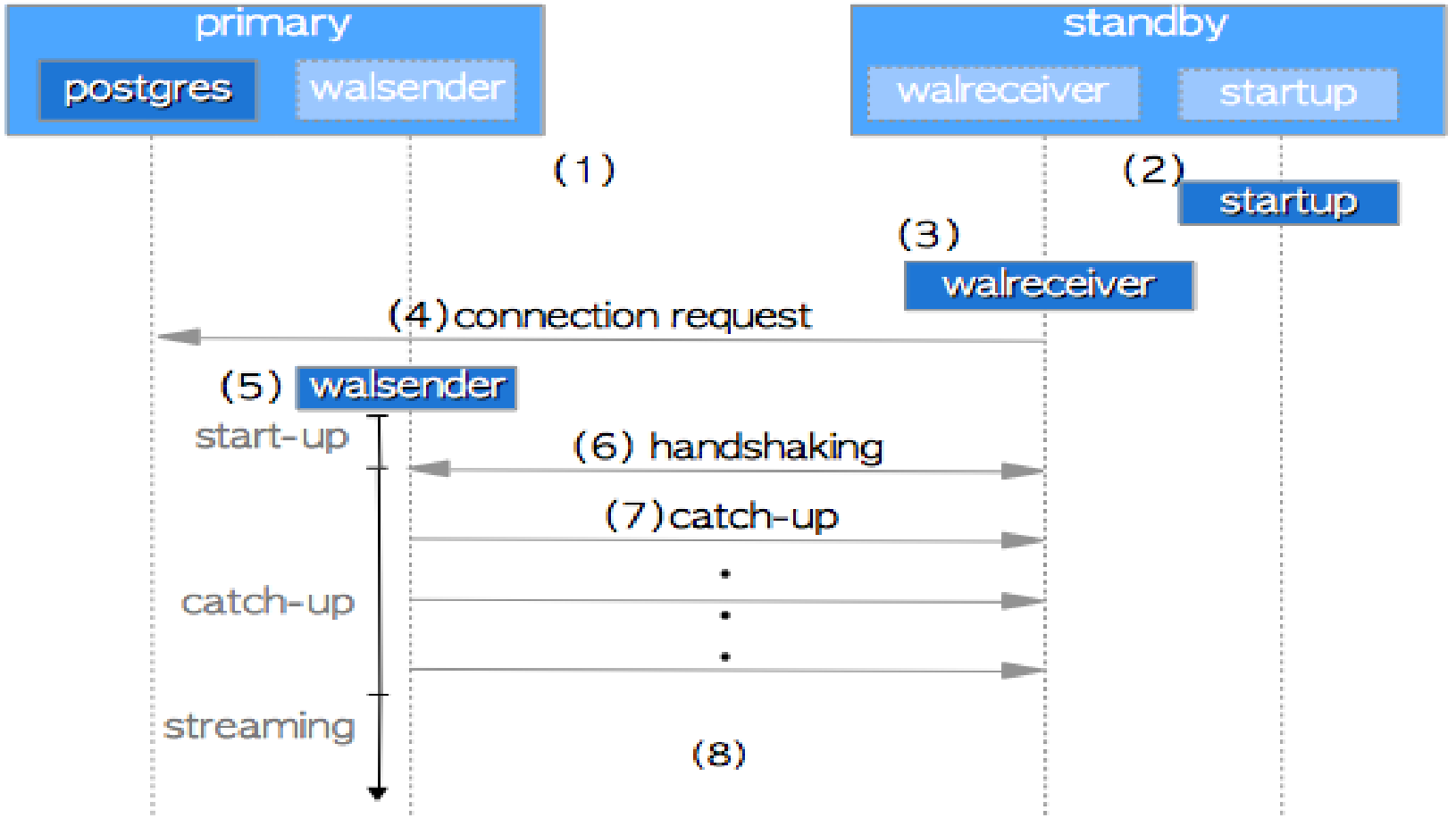
同步复制的实现逻辑



Synchronous ≠ Committed data is available on standby immediately，so also can cause replication lag example network delay、replica too busy

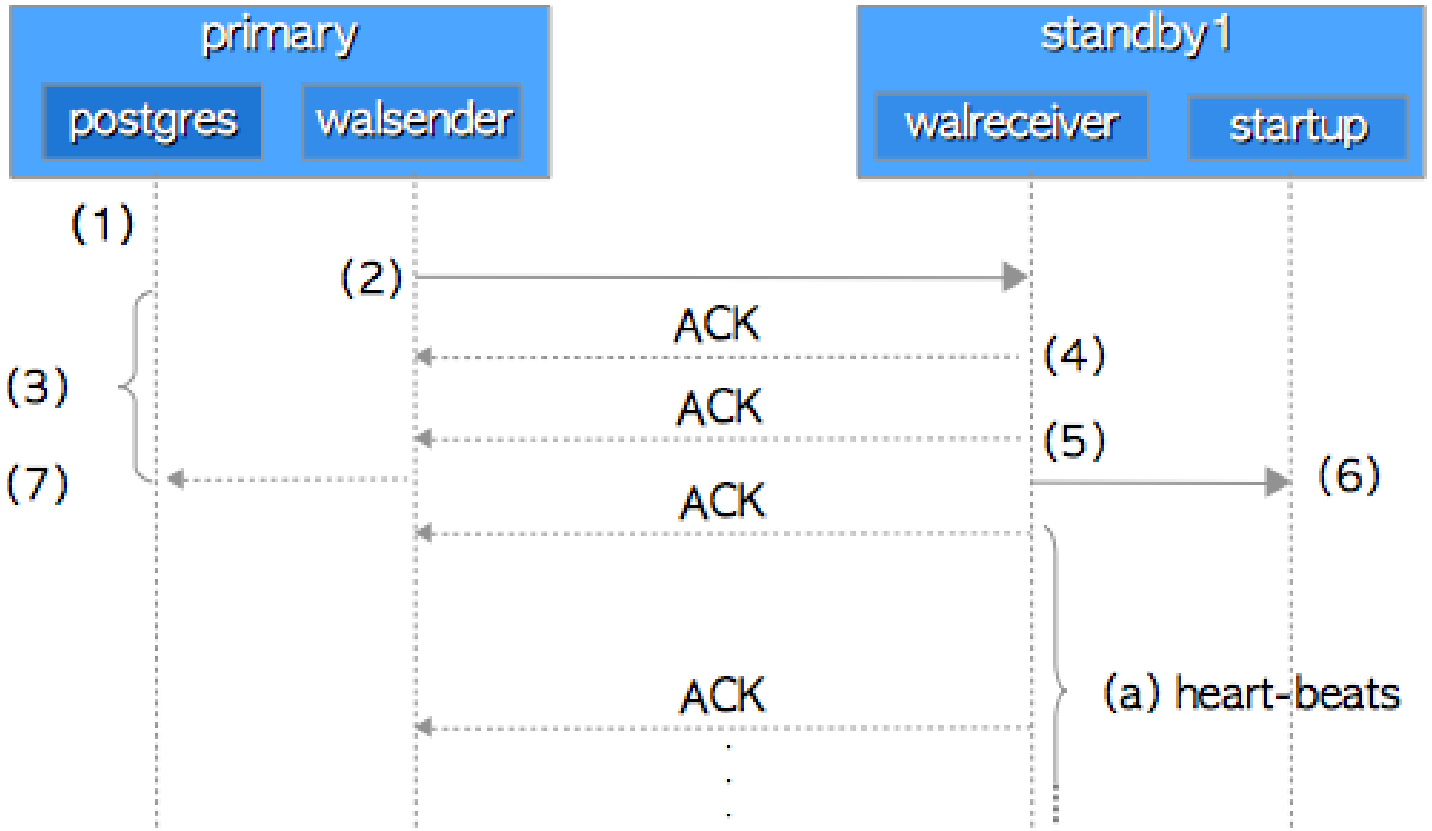
流复制-Internal architecture

流复制启动过程



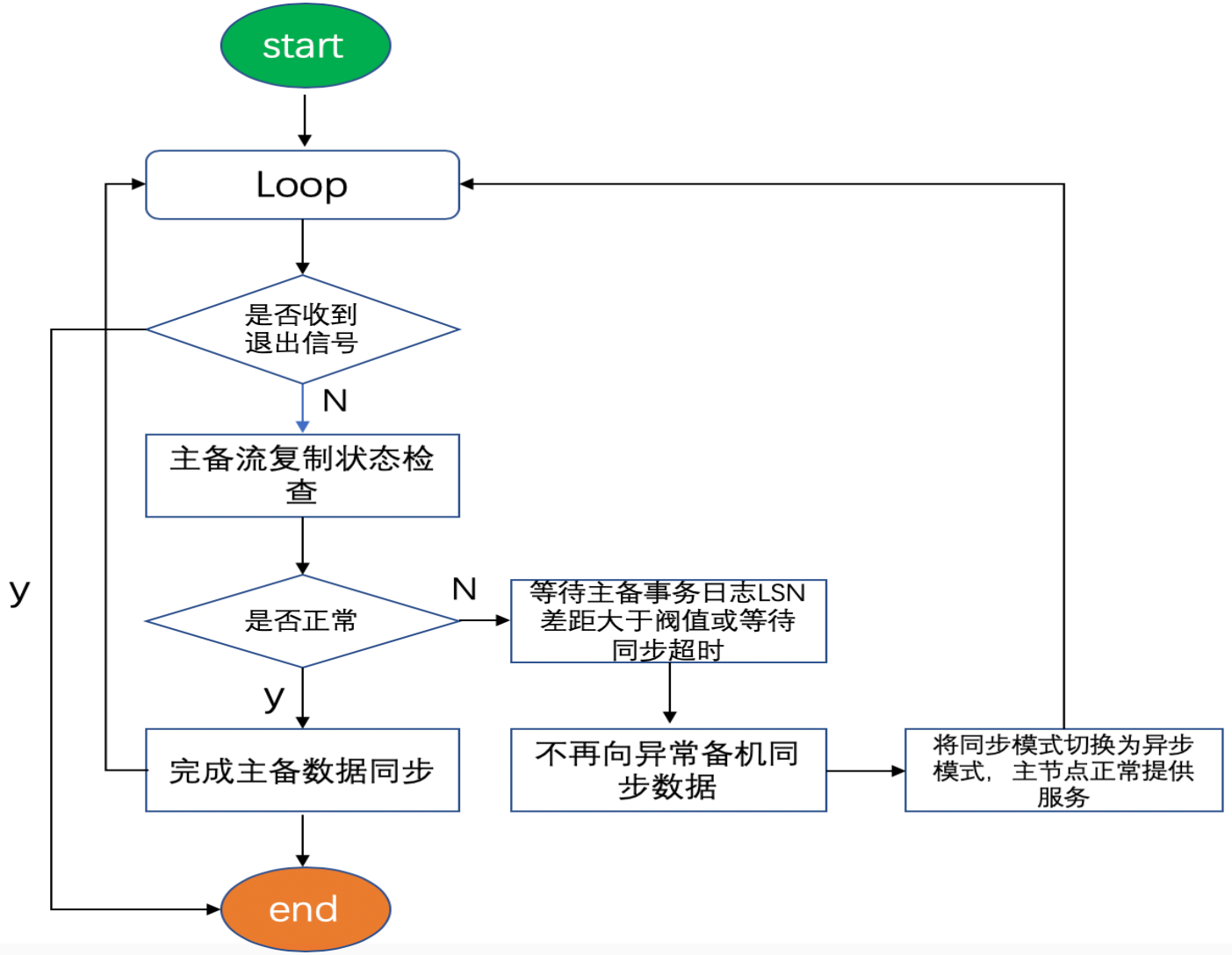
流复制-Internal architecture

流复制通信过程

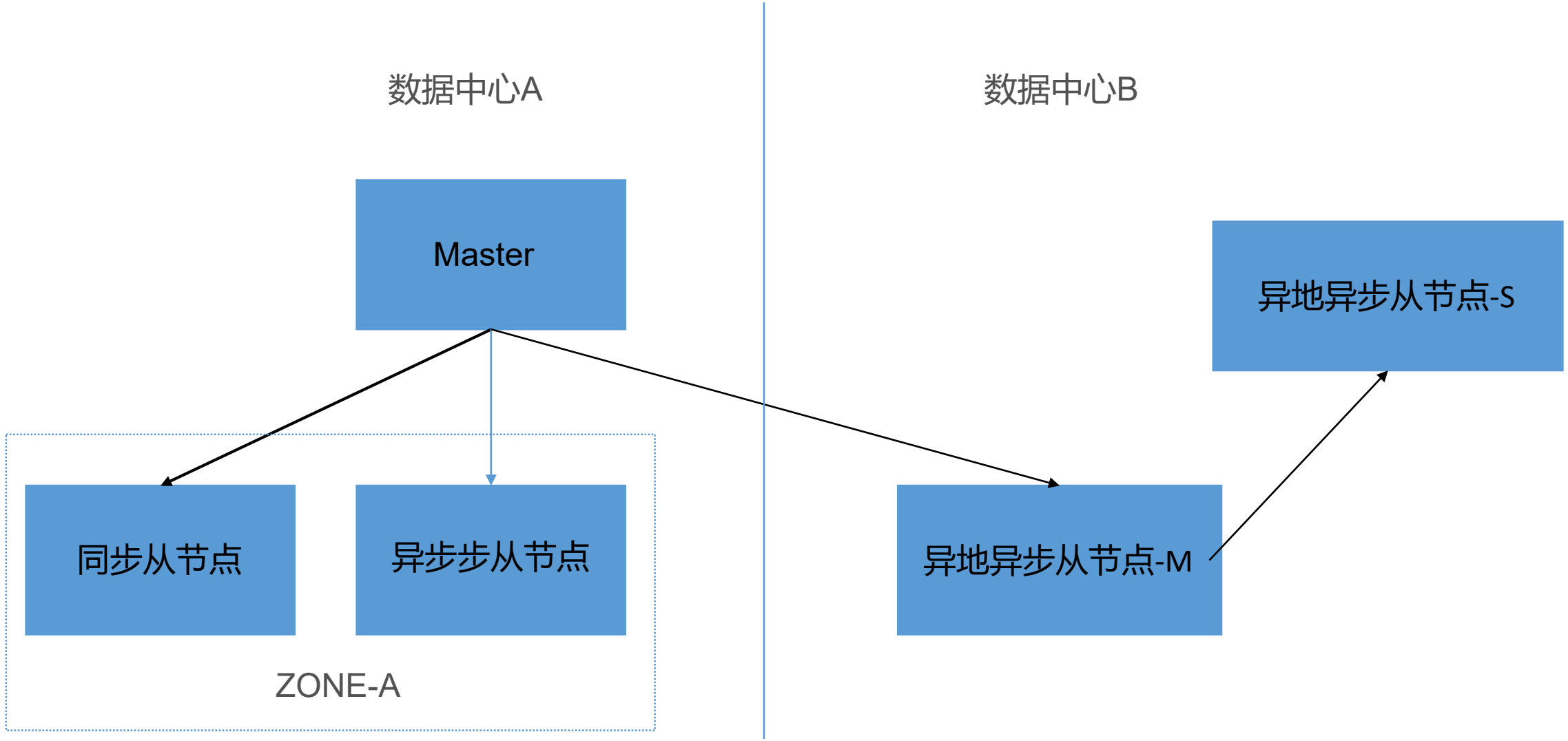


AntDB的扩展实现-自适应切换

解决同步从节点宕机后，造成主节点hang住无法为业务提供服务，影响了业务的持续高可用特性问题。

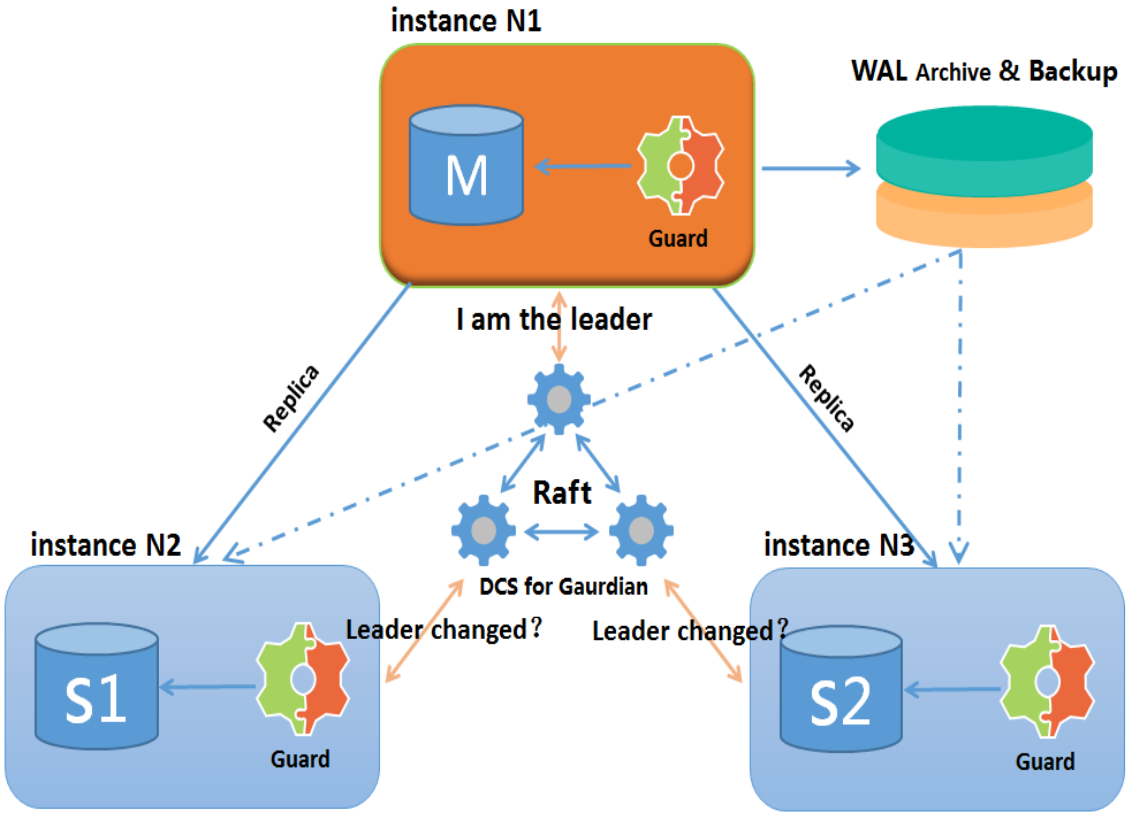


AntDB的扩展实现-按域自动切换



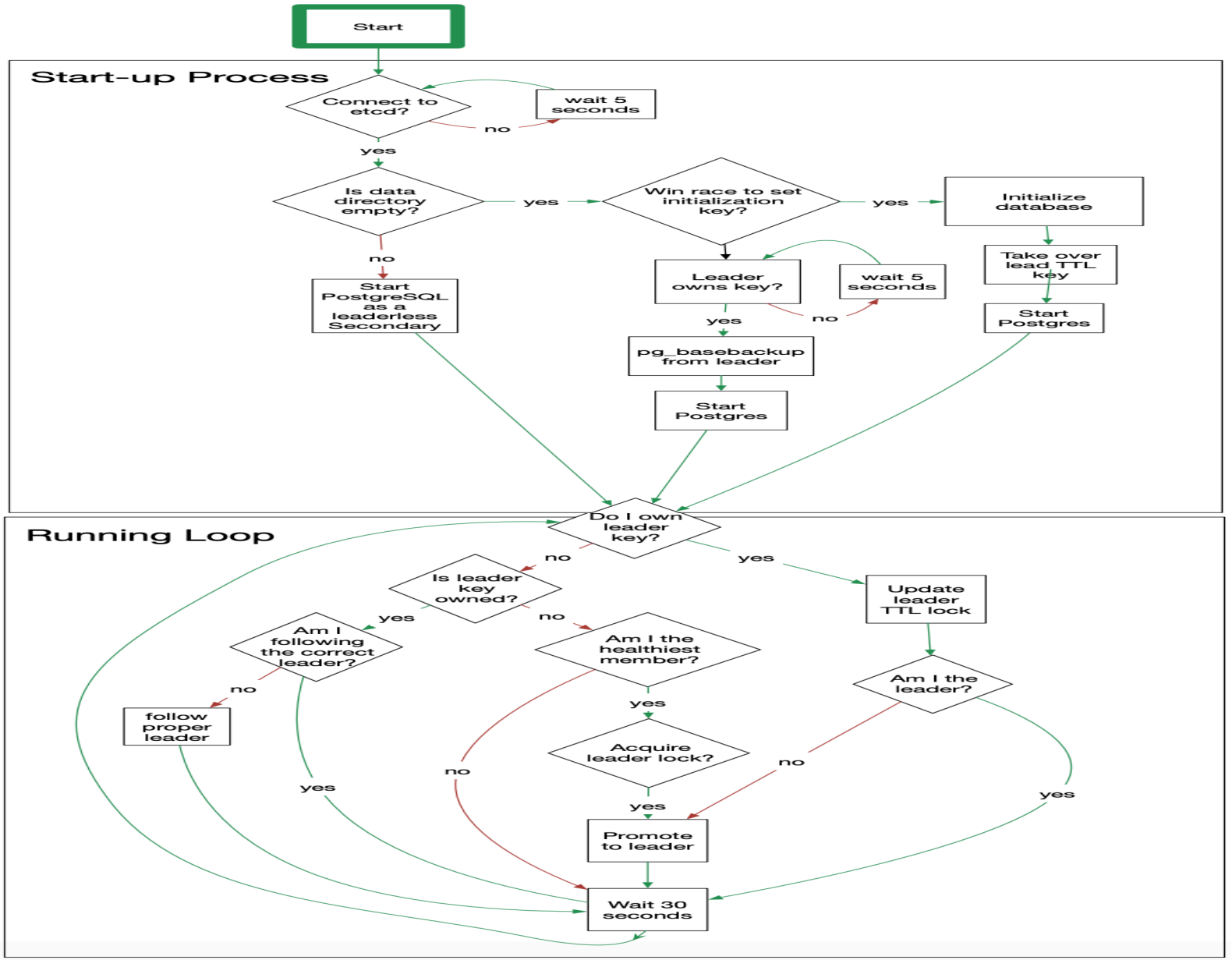
选取同zone下LSN最大备节点提升为新主节点

AntDB的扩展实现-自动高可用



- **自主创新的Guard工具结合分布式一致性Raft协议完成集群状态同步，故障自动切换**
- **秒级切换：**数据无丢失
- **支持同步和异步两种模式，**且支持同步和异步模式的自适应切换
- **支持不同级别的同步复制：** remote_write ， on， remote_apply ， 应对不同的数据安全和性能要求

AntDB的扩展实现-自动高可用



Guard进程会有两个状态Start-up Process和Running Loop，不断的check集群的状态进行leader的选举

流复制相关工具

监控 pg_stat_replication系统表记录了主从状态信息

```
postgres=# select * from pg_stat_replication ;
-[ RECORD 1 ]----+------------------------------
pid              | 14757
usesysid         | 10
usename          | danghb
application_name | db1_2
client_addr      | 10.21.20.175
client_hostname  |
client_port      | 18832
backend_start    | 2019-05-16 15:45:21.7851+08
backend_xmin     |
state            | streaming
sent_lsn         | 2/D55A1748
write_lsn        | 2/D55A1748
flush_lsn        | 2/D55A1748
replay_lsn       | 2/D55A1748
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 1
sync_state       | sync
```

Replication connection information
Standby IP address, Port number,
ROLE for replication、Replication start time, etc

Replication progress
How far has master sent WAL?
How far has standby written, flushed or replayed WAL?

Replication status
Current synchronous mode
Standby has already caught up with master?

流复制相关工具

管理接口

pg_wal_replay_pause()，pg_is_wal_replay_paused()，pg_wal_replay_resume()
pg_last_xact_replay_timestamp()， pg_last_wal_replay_lsn()

```
[root@localhost ~]# psql -h 192.168.79.134 -p 6432 -U postgres
DEBUG:  CommitTransaction
DEBUG:  name: unnamed; blockState:       STARTED; state: INPROGR, xid/subid/cid: 0/1/0, nestlvl: 1, children:
psql (9.3.4)
Type "help" for help.

postgres=# select pg_is_xlog_replay_paused()
postgres-# ;
DEBUG:  StartTransactionCommand
DEBUG:  StartTransaction
DEBUG:  name: unnamed; blockState:       DEFAULT; state: INPROGR, xid/subid/cid: 0/1/0, nestlvl: 1, children:
DEBUG:  CommitTransactionCommand
DEBUG:  CommitTransaction
DEBUG:  name: unnamed; blockState:       STARTED; state: INPROGR, xid/subid/cid: 0/1/0, nestlvl: 1, children:
LOG:  duration: 216.251 ms  statement: select pg_is_xlog_replay_paused()
;
 pg_is_xlog_replay_paused
--------------------------
 f
(1 row)

postgres=#
```

PostgreSQL社区南京活动交流群

社区QQ交流群

南京大会微信交流群

Thank you.