PostgreSQL High Availability

Master-Slave(s)-PITR

罗翼 @Qunar.com Search Team

2011.06.26

去哪儿? Qunar.Com 聪明你的旅行

大纲

- 基本概念
- PostgreSQL 9.0 的 HA 结构概览
- Stream-Replication 配置
- PITR 配置
- FAQ
- That's Not All
- Stream-Replication 未来

去哪儿? Qunar.Com 聪明你的旅行

基本概念

- Master
  - 主库，可读可写
- Slave
  - 从库，不可写
  - 不可读模式——StandBy
  - 可读模式——Stream Replication
  - trigger_file 可让 Slave 变成 Master
- PITR
  - Point-In-Time Recover
  - 通过 Base Backup 和 WAL 文件，将数据库恢复到 Base Backup 以后的任意时间点

基本概念

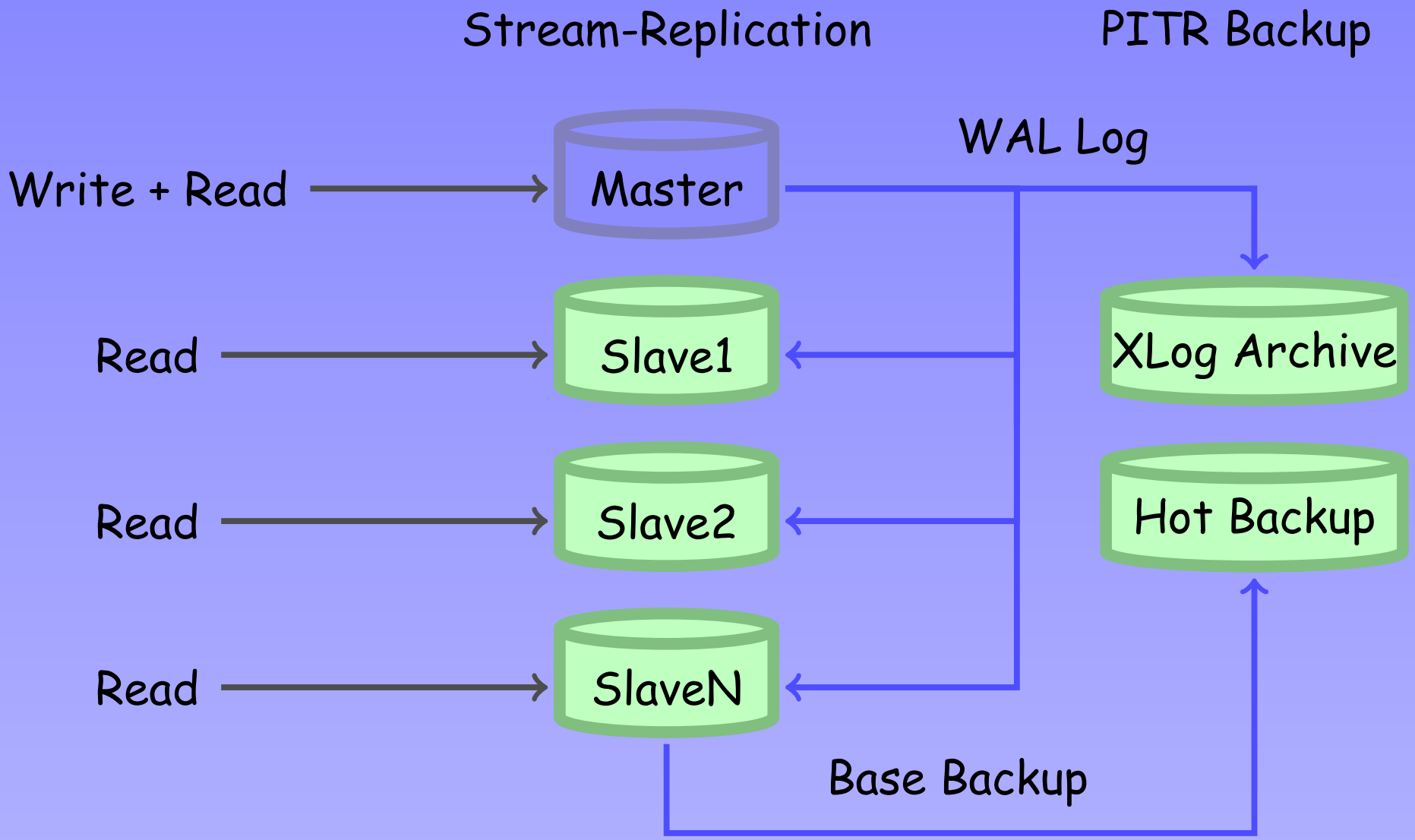
- WAL
  - WAL = Write Ahead Log （MySQL BinLog ?）
  - PostgreSQL 在对真正的数据库做任何写操作之前，都会将相关的信息记录到 WAL 文件中
  - 仅仅根据 WAL 文件，即可 Replay PostgreSQL 的所有数据写操作——并且是带事务保护的
  - 必须按照时序 Replay WAL 文件

- archive_command
  - 每生成一次完整的 WAL 文件，PostgreSQL 会调用 archive_command ，用户可以配置该变量，定制化自己的 WAL 分发策略。

- restore_command
  - 以 Slave 模式启动时，PostgreSQL 会不停地调用 restore_command 获取下一步所需要的 WAL 文件。

去哪儿? Qunar.Com 聪明你的旅行

PostgreSQL 9.0 的 HA 结构概览



去哪儿? Qunar.Com 聪明你的旅行

PostgreSQL 9.0 的 HA 结构概览

- 所有的写请求，被发送到 Master
- 所有的读请求，被均衡到 Master + 各个 Slave
- 收集 Master 产生的所有 WAL 文件，做为 PITR 的 XLog 恢复数据
- 定期对 Slave 做 Base Backup，便于快速恢复到需要的时间点

Stream-Replication 配置 Master

- pg_hba.conf 认证配置
  - Stream-Replication 使用名为 replication 的特殊用户
  - 该用户对所有数据库有 trust 权限
  - 主要根据 IP 来限制该用户的访问

```
host    replication    all    192.168.128.124/32    trust
```

Stream-Replication 配置 Master

- postgresql.conf 配置
  - 确保监听在 client 能访问的 IP 地址而不是 Unix Socket 上

```
listen_addresses = '*'
```

  - 设定 wal_level

```
wal_level = hot_standby
```

  - 设定 max_wal_senders，限定 slave 连接的上限

```
max_wal_senders = 2
```

  - 打开 archive_mode

```
archive_mode = on
```

Stream-Replication 配置 Master

- archive_command 配置

```
archive_command =
/usr/bin/omnipitr-archive
-l /omnipitr/omnipitr.log
-s /omnipitr
-dr gzip=rsync://r_ip/master_PITR/
-dl gzip=/l_dir/master_PITR
-db /omnipitr/dstbackup
--pid-file /omnipitr/omnipitr.pid
-v "%p"
```

```
# 用 omnipitr-archive 做 WAL 分发
# omnipitr-archive 的 LOG 文件路径
# omnipitr-archive 的 STATE 目录
# 用 rsync 将 WAL 文件同步到 slave 机器
# 本地的 WAL 文件目录
# 暂时不知道用途
# pid 文件，防止多个实例
# 啰嗦的 LOG
```

- omnipitr-archive 的文档

```
perldoc /usr/share/doc/omnipitr/omnipitr-archive.pod
```

去哪儿? Qunar.Com 聪明你的旅行

Stream-Replication 配置 Slave

- pg_hba.conf 无特殊配置
- postgresql.conf 无特殊配置，但 ……
  - 将 archive_mode 设为 off ，防止 XLog 文件互相污染
- recovery.conf 配置
  - PostgreSQL 启动时，如果 $PGDATA 目录下有 recovery.conf 文件，则自动进入 slave 模式
  - 默认的 slave 模式是 StandBy 模式，不能响应 ReadOnly 的查询
  - 如果 recovery.conf 中有 primary_conninfo，则进入 Stream-Replication，能响应 ReadOnly 的查询
  - 在 Stream-Replication 模式下，slave 会从 walrecv 进程和 restore_command 分别读入需要的 WAL 文件

Stream-Replication 配置 Slave

- recovery.conf 配置
  - standby_mode

```
standby_mode = 'on'
```

  - trigger_file: 将 Slave 变成 Master

```
trigger_file = '/export/pgdata/slave.trigger'
```

  - primary_conninfo

```
primary_conninfo =
'host=192.168.12.104 port=5432
user=abc password=xyz'
```

Stream-Replication 配置 Slave

- restore_command 配置

```
restore_command =
/usr/bin/omnipitr-restore
-l /omnipitr/omnipitr.log
-s gzip=/l_dir/PITR_FILES/
-f /p_dir/slave.trigger
-r
-p /var/run/omnipitr/removal.stop
-sr
-v
```

```
# 用 omnipitr-restore 做 WAL 恢复
# omnipitr-archive 的 LOG 文件路径
# 本地 WAL 文件目录
# slave 的 trigger_file
# 移除不必要的 WAL 文件 (remove)
# 如果存在，将不会继续删除 WAL 文件
# Stream-Replication 标志
# 啰嗦的 LOG
```

- omnipitr-restore 的文档

```
perldoc /usr/share/doc/omnipitr/omnipitr-restore.pod
```

去哪儿? Qunar.Com 聪明你的旅行

Stream-Replication 配置 LOG & CHECK

- Master 上的 wal sender

```
ps axuww |grep postgres |grep sender
postgres:  wal sender process ...  streaming 1A/EA0370D0
```

- Slave 上的 wal receiver

```
ps axuww |grep postgres |grep receiver
postgres:  wal receiver process streaming 1A/E909E788
```

Stream-Replication 配置 LOG & CHECK

- Master 上的 omnipitr-archive LOG

```
tail -f /var/log/omnipitr.log
omnipitr-archive : LOG : Timer [Compressing with gzip] took: 0.706s
omnipitr-archive : LOG : Timer [Sending x.gz to rsync://r_ip/xx.gz] took: 0.128s
omnipitr-archive : LOG : Segment xxx successfully sent to all destinations.
```

- Slave 上的 omnipitr-restore LOG

```
tail -f /var/log/omnipitr.log
omnipitr-restore : LOG : Called with parameters: xxx
omnipitr-restore : FATAL : Requested file does not exist, and it is streaming
                           replication environment. Dying.
```

Stream-Replication 配置 LOG & CHECK

- Slave 上的 pg_log 启动 LOG

```
tail -f /export/pgdata/pg_log/pgrun_xxx.log
LOG: entering standby mode
LOG: redo starts at 10/C7070CD0
LOG: consistent recovery state reached at 10/CE039210
LOG: database system is ready to accept read only connections
LOG: invalid record length at 10/D1019198
LOG: streaming replication successfully connected to primary
```

去哪儿? Qunar.Com 聪明你的旅行

PITR 配置 选择

- PITR = Base Backup + WAL
- Master 的 WAL 文件已经全部收集了
- 在 Master 上做 Base Backup
  - 读取 $PGDATA 中所有的文件，会击溃 OS 的 FS Cache
  - TAR + GZIP 消耗大量的 CPU 资源
  - 稳定，权限，性能 —— 谁愿意给 Master 再多加一个任务呢？
- 在 Slave 上做 Base Backup
  - 仍然需要到 Master 执行 pg_start_backup
  - 需要等待主库的 CHECKPOINT，会消耗更多时间
- 当然还是选择 Slave ！

PITR 配置 备份

- 在 Slave 机器上运行 omnipitr-back-slave

```
/usr/bin/omnipitr-backup-slave
-D /export/pgdata
-h master.db.qunar.com
-U abc -d testdb -P 5432
-s gzip=/l_dir/PITR_FILES/
-dl gzip=/l_dir/hot_backup
-l /omnipitr/omnipitr.log
--pid-file /omnipitr/omnipitr-bs.pid
-p /omnipitr/removal.stop
-cm
-v
--psql-path /opt/pg90/bin/psql
--tar-path /opt/tar/bin/tar
-nn
```

```
# 用 o-b-s 做 Base Backup
# $PGDATA 目录
# Master 地址
# Master 连接信息
# Master WAL 目录
# 生成的 Base Backup 文件存放目录
# 运行 LOG 目录
# pid 文件, 防止多实例同时运行
# 停止删除旧的 WAL 文件的标志文件
# CALL MASTER pg_start_backup
# 啰嗦 LOG
# PSQL 可执行文件的路径
# >1.30 版本的 GNU TAR
# 不掉用 nice 改变运行优先级
```

PITR 配置 恢复

- 解压缩 data 和 xlog 文件

```
ls -alh *.tag.gz
xxx 1.1G Jun 25 22:53 masterdb-data-2011-06-25.tar.gz
xxx 307M Jun 25 22:55 masterdb-xlog-2011-06-25.tar.gz
```

- 将当前收集到的所有的 WAL 文件放入 pg_xlog 文件夹
- 调用 pg_ctl 启动数据库
  - 如果有 recovery.conf 文件存在，则自动进入 slave 模式

PITR 配置 LOG & CHECK

- omnipitr-back-slave 的 LOG

```
tail -f /var/log/omnipitr/omnipitr.log
LOG : Timer [SELECT pg_start_backup('xxx')] took: 27.316s
LOG : pg_start_backup('omnipitr') returned 10/E1000020.
LOG : Timer [select pg_read_file( 'backup_label', 0, (
      pg_stat_file( 'backup_label' ) ).size )] took: 0.012s
2011-06-25 22:45:50.572652 LOG : Waiting for checkpoint (based
      on backup_label from master) - CHECKPOINT LOCATION: 10/E1070BE8
2011-06-25 22:50:45.932004 LOG : Checkpoint .
LOG : Starting "gzip" writer to /l_dir/masterdb-data-2011-06-25.tar.gz
LOG : Timer [Compressing $PGDATA] took: 190.923s
LOG : Timer [SELECT pg_stop_backup()] took: 3.015s
LOG : pg_stop_backup() returned 10/E9081CB0.
LOG : Timer [Making data archive] took: 516.659s
```

- 请注意 Waiting for checkpoint 的时间长度

PITR 配置 计划

- Base Backup 和恢复时间之间，如果有巨量的 WAL 文件，会导致 recover 过程很慢 (小时级别)
- 所以需要定期的 Base Backup
- 制定 Base Backup 的时间表以后，定期删除过期的 WAL 文件

去哪儿? Qunar.Com 聪明你的旅行

FAQ

- 过程中出现莫名其妙的错误
  - 检查牵涉的所有目录是否都是 `postgres` 用户的 owner，并且权限 mask = 0600
- 修改了配置文件的参数，不起作用
  - 用 `pg_ctl` restart 数据库
- Master 的 WAL 文件无法传输到 Slave
  - 确保从库的 `rsyncd.conf` 文件内容正确——特别是权限！
- `omnipitr-back-slave` 跑了好长时间，没结束啊
  - 确保 `omnipitr-back-slave` 和 `omnipitr-archive` 有共同的压缩方案（gzip,bzip2...）

去哪儿? Qunar.Com 聪明你的旅行

FAQ

- 参数都对，主从还是连不上
  - 确保 Slave 的 `max_connections` 参数的值大于等于主库的该参数的值
- 对 Slave 做 `pg_dump` 总是失败，报个莫名其妙的错误

```
max_standby_archive_delay = 1800s
max_standby_streaming_delay = 1800s
```

去哪儿? Qunar.Com 聪明你的旅行

That's Not All

- 一个额外的 monitor-system 来确认 Master die 并且 trigger Slave
  - 由于应用的场景各异，限制各不相同，无法做出通用的 FailOver System
- 应用通过 VIP 或者 DNS 的方式，动态选择 Master
- Slave FailOver 以后，如何接入新的 Slave，以及新的 PITR 过程

去哪儿? Qunar.Com 聪明你的旅行

Stream-Replication 未来 9.1 , 9.2

- 同步事务（可能并不如你想象中那么“同步”）
- UNLOGGED TABLE （某些表并不需要同步）
- 级联 WAL Sender ，减少 Master 的负载
- Slave 同步 Master 某段时间之前的数据（一个永远处于一周前的 Slave）
- 丢掉幻想。有些需求，是二进制模式同步永远无法解决的！ bin-mode 和 trigger-mode 需要并存