第六章
备份、恢复、容灾

digoal

目标

- 1、掌握数据库的持续备份、任意时间点恢复，导入、导出，容灾。

目录

- pg_dump
  - 结构、数据、库级、表级，可定制owner、权限、表空间
- pg_dumpall
  - 全局元信息（用户、表空间）
- DB 端copy
- 客户端copy(copy协议)
- copy single byte char 元符号使用
  - https://github.com/digoal/blog/blob/master/201805/20180510_01.md

例子

- 导出用户、表空间定义
  - pg_dumpall -g
- 导出postgres库结构
  - pg_dump -F p -f /tmp/dump.csv -s -h 127.0.0.1 -p 12000 -U postgres -d postgres
- 导出postgres库t1,t2表数据
  - pg_dump -F p -f /tmp/dump.csv -t t1 -t t2 -h 127.0.0.1 -p 12000 -U postgres -d postgres
- 服务端导出，导出到数据库所在服务器
  - postgres=# copy t1 to '/tmp/t1.dmp';
- 客户端导出，导出到psql命令所在服务器
  - postgres=# \copy t1 to '/tmp/t1.dmp'
  - psql -c "copy t1 to stdout" > /tmp/t1.dmp
- single char使用
  - postgres=# copy aa from '/home/digoal/aa.csv' with (delimiter U&'\0009');
  - postgres=# copy aa from '/home/digoal/aa.csv' with (delimiter E'\t');

目录

- 库级一致snapshot
- 事务隔离级别
  - RR

目录

- snapshot一致性视角

- 所有worker，导入同一snapshot
  - begin transaction isolation level repeatable read;
  - postgres=# SELECT pg_export_snapshot();
  - pg_export_snapshot
  - ----------
  - 000007E9-1
  - (1 row)

  - begin transaction isolation level repeatable read;
  - SET TRANSACTION SNAPSHOT '000007E9-1';

  - 又比如
  - pg_dump --snapshot=00000736-1

目录

- pg_restore
- copy

- 例子
- pg_restore -d postgres -h 127.0.0.1 -p 12000 -U postgres /tmp/t1.csv
- psql
  - copy t1 from '/home/pg12/t1.csv'

目录

- https://github.com/digoal/blog/blob/master/201204/20120412_01.md
- pg_restore -f ./digoal.list -F c -l ./digoal.dmp
- 应用
  - 调整还原顺序
  - 注释不需要还原的对象

2. 2878; 1262 16386 DATABASE - digoal postgres 这行的意思

```
2878 对应 dumpId
1262 对应 catalogId.tableoid
16386 对应 catalogId.oid
DATABASE 对应 desc
- 对应 te->namespace ? te->namespace : "-"
digoal 对应 tag
postgres 对应 owner
```

```
;
; Archive created at Thu Apr 12 09:32:27 2012
;     dbname: digoal
;     TOC Entries: 126
;     Compression: -1
;     Dump Version: 1.12-0
;     Format: CUSTOM
;     Integer: 4 bytes
;     Offset: 8 bytes
;     Dumped from database version: 9.1.3
;     Dumped by pg_dump version: 9.1.3
;
;
; Selected TOC Entries:
;
2878; 1262 16386 DATABASE - digoal postgres
7; 2615 25070 SCHEMA - digoal digoal
5; 2615 2200 SCHEMA - public postgres
2879; 0 0 COMMENT - SCHEMA public postgres
2880; 0 0 ACL - public postgres
```

目录

- 全量备份包含哪些文件
  - datafile
  - tablespace
  - wal
- 手动
  - pg_start_backup
    - ckpt、强制开启FPW
  - copy file
  - pg_stop_backup
- 自动
  - pg_basebackup
- 有效备份集：
  - 数据文件（datafile, 自定义tbs）
  - WAL(检查点逻辑位点 ~ wal文件位点)

例子

- 1、创建REPLICATION角色
- 2、配置pg_hba.conf，允许远程流式备份
- 3、远程使用pg_basebackup备份
  - pg_basebackup -F t -c fast -X stream -D /tmp/pg12bak -h 127.0.0.1 -p 12000 -U postgres

目录

- pg_rman原理
- page head, lsn
- https://github.com/digoal/blog/blob/master/201608/20160826_01.md

目录

- 存储快照
  - 云盘快照
- 逻辑卷快照
  - lvm
- 文件系统快照
  - zfs
- 一致性
  - start backup
  - stop backup
  - 以及所有相关wal

目录

- $PGDATA/pg_wal/archive_status
  - .ready
  - .done
- archive_mode
  - always
  - on
  - off
- archive_command = '' # command to use to archive a logfile segment
- # placeholders: %p = path of file to archive
- # %f = file name only
- # e.g. 'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/archivedir/%f'
- #restore_command = '' # command to use to restore an archived logfile segment
- # placeholders: %p = path of file to restore
- # %f = file name only
- # e.g. 'cp /mnt/server/archivedir/%f %p'
- # (change requires restart)

目录

- 1、创建replication角色
- 2、配置pg_hba.conf允许远程流复制连接
- 3、使用pg_receivexlog流式实时接收REDO

目录

- 在线逻辑备份
- 逻辑备份一致性介绍
- 逻辑备份的并行
- 逻辑恢复
- TOC介绍
- 在线全量备份
- 在线BLOCK级增量备份
- 在线快照级全量备份
- 归档
- 持续归档

- PITR（时间点恢复）介绍
- 基于全量+归档的PITR
- 基于pg_rman全量+BLOCK增量+归档的PITR
- 基于快照+归档的PITR
- 实时容灾
- 时间线介绍
- flashback介绍(pg_rewind)
- 时间线错乱修复
- 练习

- # - Recovery Target -
- # Set these only when performing a targeted recovery.
- #recovery_target = '' # 'immediate' to end recovery as soon as a
- # consistent state is reached
- #recovery_target_name = '' # the named restore point to which recovery will proceed
- #recovery_target_time = '' # the time stamp up to which recovery will proceed
- #recovery_target_xid = '' # the transaction ID up to which recovery will proceed
- #recovery_target_lsn = '' # the WAL LSN up to which recovery will proceed
- #recovery_target_inclusive = on # Specifies whether to stop:
- # just after the specified recovery target (on)
- # just before the recovery target (off)
- #recovery_target_timeline = 'latest' # 'current', 'latest', or timeline ID
- #recovery_target_action = 'pause' # 'pause', 'promote', 'shutdown'

目录

例子

- 1、准备目标环境（与源库兼容的硬件平台、操作系统平台、足够的空间）
- 2、准备数据库二进制，以及所有依赖的插件
- 3、拷贝数据文件到目标目录
- 4、配置postgresql.conf
  - port=xxxx
  - hot_standby=on
- 5、配置recovery.conf
  - restore_command='cp /data01/digoal/wal/%f %p'
  - recovery_target_time='2019-05-04 13:33:56.438883+08'
  - recovery_target_timeline='latest'
  - recovery_target_action='pause'
- 6、启动数据库
- 7、拷贝需要的归档到目标目录，可以看日志或pg_is_wal_replay_paused()判断是否已达到目标位置
- 8、等待数据库恢复到目标位置

目录

- https://github.com/digoal/blog/blob/master/201608/20160829_03.md

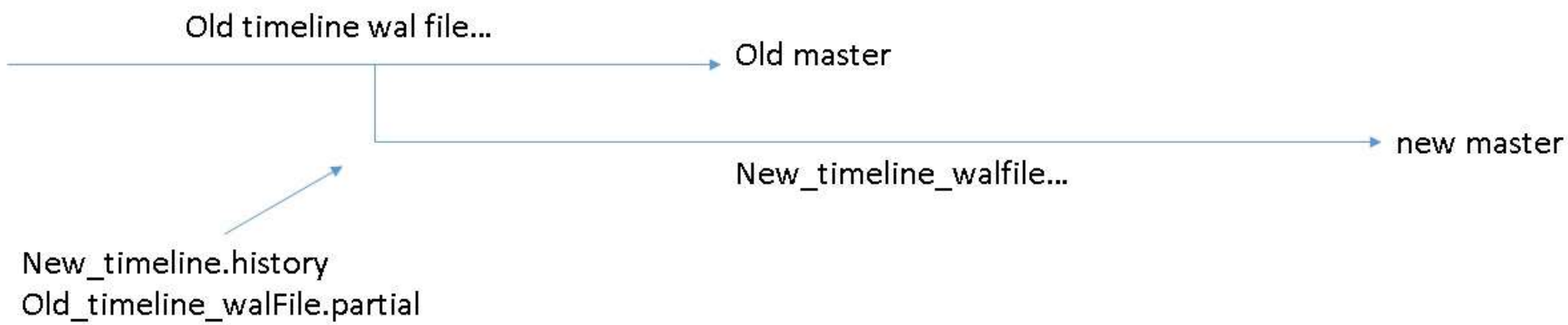
目录

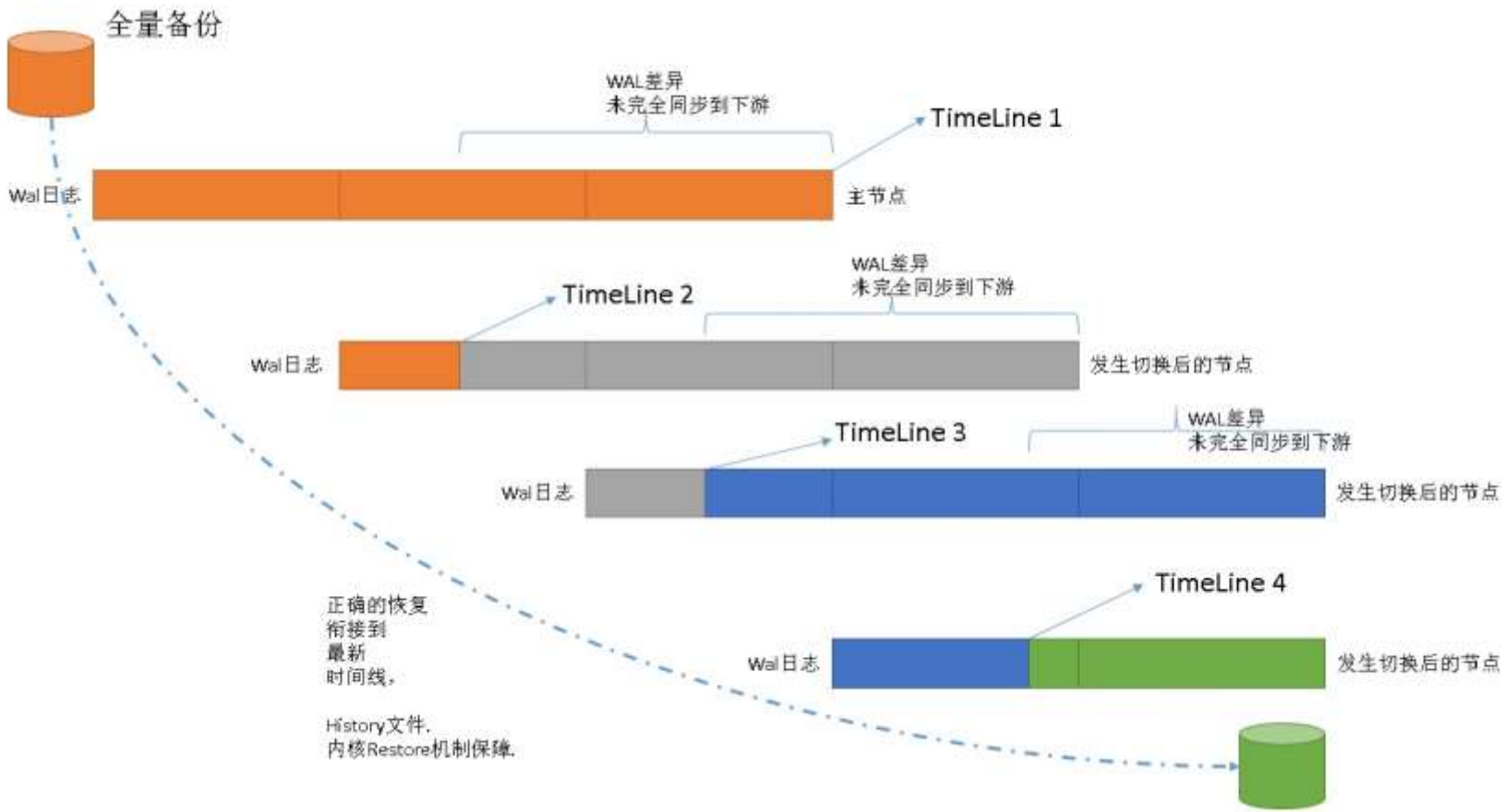
- 以ZFS为例
- 1、找到离目标时间点最近的快照
- 2、克隆快照
- 3、配置postgresql.conf
  - port=xxxx
  - hot_standby=on
- 4、配置recovery.conf
  - restore_command='cp /data01/digoal/wal/%f %p'
  - recovery_target_time='2019-05-04 13:33:56.438883+08'
  - recovery_target_timeline='latest'
  - recovery_target_action='pause'
- 5、启动数据库
- 6、拷贝需要的归档到目标目录，可以看日志或pg_is_wal_replay_paused()判断是否已达到目标位置
- 7、等待数据库恢复到目标位置

目录

- 1、构建异地standby
  - recovery.conf
    - restore_command='cp /data01/digoal/wal/%f %p' # 拷贝需要的WAL到wal目录
    - recovery_target_timeline = 'latest'
    - standby_mode = on
    - primary_conninfo = 'host=主库IP port=端口 user=用户 password=密码'
- 2、standby 定期快照(例如ZFS snapshot)
- 3、处理时间线错乱
  - 回退快照(snapshot回退)
  - 重启STANDBY

目录


Old timeline wal file...
Old master
new master
New_timeline_walfile...
New_timeline.history
Old_timeline_walFile.partial


全量备份
WAL差异
未完全同步到下游
TimeLine 1
Wal日志
主节点
TimeLine 2
WAL差异
未完全同步到下游
Wal日志
发生切换后的节点
TimeLine 3
WAL差异
未完全同步到下游
Wal日志
发生切换后的节点
TimeLine 4
Wal日志
发生切换后的节点
正确的恢复
衔接到
最新
时间线，
History文件，
内核Restore机制保障。

目录

pg_rewind

- https://www.postgresql.org/docs/11/app-pgrewind.html
- https://github.com/digoal/blog/blob/master/201901/20190128_02.md
- **原理与修复步骤**
- 1、使用pg_rewind功能的前提条件：必须开启full page write，必须开启wal hint或者data block checksum。
- 2、需要被修复的库：从激活点开始，所有的WAL必须存在pg_wal目录中。如果WAL已经被覆盖，只要有归档，拷贝到pg_wal目录即可。
- 3、新的主库，从激活点开始，产生的所有WAL必须存在pg_wal目录中，或者已归档，并且被修复的库可以使用restore_command访问到这部分WAL。
- 4、修改(source db)新主库或老主库配置pg_hba.conf，允许老主库通过流复制协议连接新主库。
- 5、使用pg_rewind，从新主库获得新的时间线时间点，从这个WAL位点开始解析老主库的所有WAL对应的BLOCK ID。从新主库拷贝这些BLOCK，覆盖老主库。
- 6、从老主库拷贝pg_xact 以及配置文件到需要恢复的目标实例。
- 老主库回退到切换时间点的状态。

- 7、修改被修复库(target db)的recovery.conf, postgresql.conf配置。
- 8、启动target db，连接source db接收WAL，或restore_command配置接收WAL，从切换点开始所有WAL，进行apply。
- 9、target db现在是source db的从库。

目录

- https://github.com/digoal/blog/blob/master/201901/20190128_02.md

目录

- 1、配置数据库参数，并完成一次全量备份，归档备份。
- 2、对数据库进行一些读写，记录时间，XID，创建恢复点，完成一次时间点恢复。
- 3、完成一次服务端COPY导入导出。
- 4、完成一次客户端COPY导入导出。
- 5、完成一次逻辑备份全库，并恢复到目标库。

参考

- https://www.postgresql.org/docs/11/app-pgreceivewal.html
- https://github.com/digoal/blog/blob/master/201711/20171129_02.md
- https://github.com/digoal/blog/blob/master/201708/20170812_01.md
- https://github.com/digoal/blog/blob/master/201608/20160823_09.md
- https://github.com/digoal/blog/blob/master/201608/20160823_08.md
- https://github.com/digoal/blog/blob/master/201608/20160823_07.md
- https://github.com/digoal/blog/blob/master/201608/20160823_06.md
- https://github.com/digoal/blog/blob/master/201608/20160823_05.md
- https://github.com/digoal/blog/blob/master/201608/20160823_04.md
- https://github.com/digoal/blog/blob/master/201608/20160823_03.md
- https://github.com/digoal/blog/blob/master/201608/20160823_02.md
- https://github.com/digoal/blog/blob/master/201608/20160823_01.md
- https://github.com/digoal/blog/blob/master/201204/20120412_01.md
- https://github.com/digoal/blog/blob/master/201805/20180516_03.md
- https://github.com/digoal/blog/blob/master/201805/20180510_01.md
- https://github.com/digoal/blog/blob/master/201704/20170424_05.md
- https://github.com/digoal/blog/blob/master/201901/20190129_01.md
- https://github.com/digoal/blog/blob/master/201901/20190128_02.md

资料汇总

PG进阶群

个人微信