高可用方案

1.主从

2.双主

3.多主多从

4.读写分离

5.负载均衡 sql分发

常见高可用方案

共享磁盘故障转移

　　共享磁盘故障转移避免了只使用一份数据库拷贝带来的同步开销。 它使用一个由多个服务器共享的单一磁盘阵列。

文件系统（块设备）复制

　　DRBD是用于 Linux 的一种流行的文件系统复制方案。

事务日志传送

　　温备和热备服务器能够通过读取一个预写式日志（WAL） 记录的流来保持为当前状态。如果主服务器失效， 后备服务器拥有主服务器的几乎所有数据， 并且能够快速地被变成新的主数据库服务器。这可以是同步的或异步的， 并且只能用于整个数据库服务器。

基于触发器的主-备复制

　　一个主-备复制设置会把所有数据修改查询发送到主服务器。 主服务器异步地将数据修改发送给后备服务器。当主

　　服务器正在运行时， 后备服务器可以回答只读查询。后备服务器对数据仓库查询是一种理想的选择。

　　Slony-I是这种复制类型的一个例子。它使用表粒度， 并且支持多个后备服务器。因为它

　　会异步更新后备服务器（批量）， 在故障转移时可能会有数据丢失。

基于语句的复制中间件

　　通过基于语句的复制中间件，一个程序拦截每一个 SQL 查询并把它发送给一个或所有服务器。 每一个服务器独立地操作。

　　读写查询必须被发送给所有服务器， 这样每一个服务器都能接收到任何修改。但只读查询可以被只发送给一个服务器， 这

　　样允许读负载在服务器之间分布。

异步多主控机复制

　　通过使用异步的多主控机复制， 每一个服务器独立工作并且定期与其他服务器通信来确定冲突的事务。

同步多主控机复制

　　在同步多主控机复制中，每一个服务器能够接受写请求，并且在每一个事务提交之前，被修改的数据会被从原始服务器传送给每一个其他服务器。

　　PostgreSQL不提供这种复制类型， 尽管在应用代码或中间件中可以使用PostgreSQL的两阶段提交 （PREPARE TRANSACTION和COMMIT PREPARED） 来实现这种复制。

商业方案

　　因为PostgreSQL是开源的并且很容易被扩展， 一些公司已经使用PostgreSQL并且创建了带有唯一故障转移、 复制和负载均衡能力的商业性的闭源方案。

数据分区

　　数据分区将表分开成数据集。每个集合只能被一个服务器修改。

多服务器并行查询执行

replication

逻辑复制和 物理复制

优劣和应用场景

逻辑订阅，适合于发布端与订阅端都有读写的情况。

逻辑订阅，更适合于小事务，或者低密度写（轻度写）的同步。如果有大事务、高密度写，逻辑订阅的延迟相比物理复制更高。

逻辑订阅，适合于双向，多向同步。

物理复制，适合于单向同步。

物理复制，适合于任意事务，任意密度写（重度写）的同步。

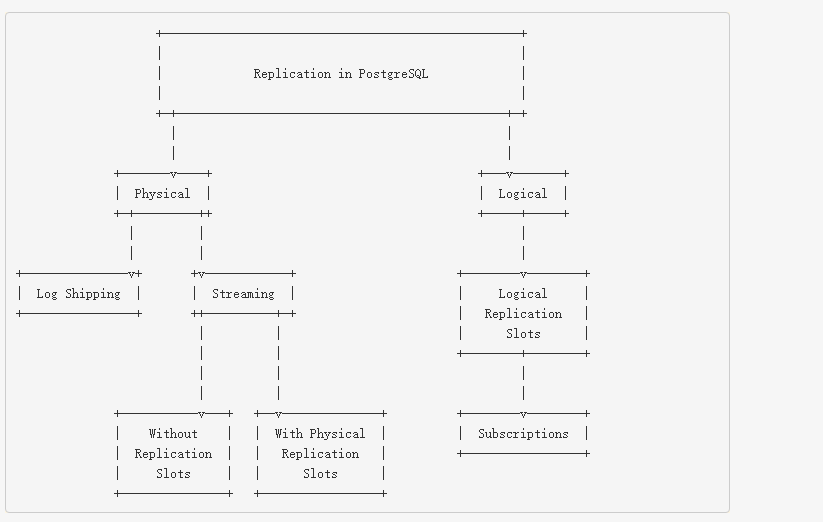
物理复制，适合于HA、容灾、读写分离。

物理复制，适合于备库没有写，只有读的场景。

实验平台 centos7+pg10 两台

总架构

postgresql高可用方案有许多第三方方案，但是今天我们主要讲pg内核自己支持的 高可用方案



物理复制，基于 wal日志

什么是wal日志？  Write-Ahead Logging

wal日志，postgresql 在做修改、插入等DDL语言时，会先将操作语句写到wal日志里；记录所有操作；然后，在check point点，replay这些日志，真正去执行这些操作

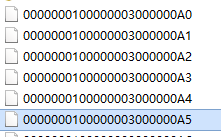
在 oracle 里 这个叫 redo日志，mysql里这个叫 binlog ,sqlserver 也叫 wal日志

wal日志 是二进制文件，默认16M(当然，可以设置）

wal文件的命名

wal日志号由64位组成，即有2\*\*64个日志编号，用完了需要重置，但是这个数量已经很多了，相当于1024P个，1024\*1024个T。

日志文件则由24个16进制数字组成，分三部分：时间线、LSN高32位、LSN低32位/（2\*\*24）的值

=

16 位 lsn=1 lsn=00000000 000000002 lsn=0000000 0000000A

20/9B0708E0

000000001 000000030 9B0708E0

如何读取wal日志。pg自带 pg\_waldump

pg\_waldump.exe --start=3/A0000001 -n 10

手动切换日志

Version>=10

select pg\_switch\_wal();

version<10  
select pg\_switch\_xlog();

每一行wal日志

rmgr : 资源名称

lsn: 0/0162D3F0 日志编号

prev 0/0162D3B8

desc ： 对日志详细信息的描述

xid 事务id

physical 复制的两种思路

准备知识

什么是归档 archive

arhive\_mode=on 开启归档进程，服务里会多一个 归档进程

当启用archive\_mode时，通过设置archive\_command将已完成的WAL段发送到归档存储

**注意：**触发归档有三种方式：

1.手动切换wal日志，select pg\_switch\_wal()

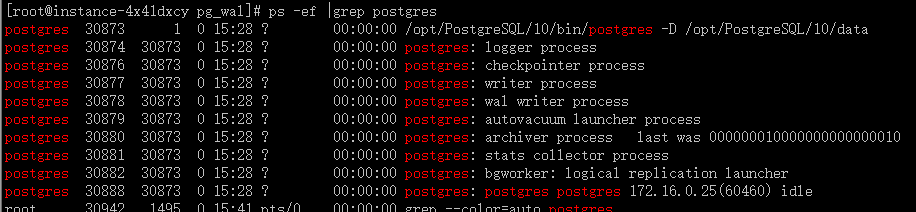
2.wal日志写满后触发归档，配置文件默认达到16M后就会触发归档，wal\_keep\_segments = 16

3.归档超时触发归档，archive\_timeout

archive\_mode=on

archive\_command=' test ! cp %p /data/%f'

特别要注意 /data 这个归档文件夹 需要有 postgres的写权限



若没有，则 archiver process 会失败；查看pg\_log可以知道原因

什么是pg\_basebackup

pg\_basebackup 是一个备份命令

用法 一则/opt/PostgreSQL/10/bin/pg\_basebackup -h 172.16.0.10 -R -D /opt/PostgreSQL/10/data -U postgres -W

-D 指定备份文件将要写入的目录

-h 要备份的主机

-R  是否写 recovery.conf

在仅添加 -D 情况下，仅仅相当于复制，除开 pid文件等 ；scp

可以在新机器上systemctl restart postgresql-10 也是可以使用的

可以对比 直接 scp data文件夹到新机器上 ，然后启动服务的效果；

（注意文件夹权限，data文件夹必须为postgres可以访问）

为什么不直接scp，而要pg\_basebackup呢；

因为pg\_basebackup 除了复制，还将主库进入备份模式，复制，再退出主库的备份模式；同时，被复制出来的data文件夹里也自动删除了 posmaster.pid

如果有 -R 参数，还会自动配置 recover.conf

什么是 recovery.conf

当数据库重启时，会检查自动redo wal日志。

在将wal日志 redo完之前，都是恢复模式

若data文件下配置了 recovery.conf 会先根据此文件中的内容做一些事情。

当standby\_mode=on 时，系统处于恢复模式。

恢复模式下，只可读不可写；可以重放wal日志。

表现一：此时若把新的wal日志copy到 pg\_wal 文件夹下，则会自动apply这个日志，实现数据库的更新

表现二：此时若 restore\_command 里有内容，则执行resotre\_command 里的命令

restore\_command 命令示例：  restore\_command ='cp /path/%f %p'

表现三：若此时

primary\_conninfo = 'user=postgres password=since2015 host=172.16.0.10 port=5432 sslmode=prefer sslcompression=1 krbsrvname=postgres target\_session\_attrs=any'

则为流复制，不断执行 流传输过来的wal日志

一：流复制

配置流复制，主库只要开启 replication 权限（pg\_hba.conf)



1. 配置 主机postgresql.conf:

wal\_level = replica # 这个是设置主为wal的主机

max\_wal\_senders = 32 # 这个设置了可以最多有几个流复制连接，差不多有几个从，就设置几个 wal\_keep\_segments = 256 ＃ 设置流复制保留的最多的xlog数目

wal\_sender\_timeout = 60s ＃ 设置流复制主机发送数据的超时时间

max\_connections = 100 # 这个设置要注意下，从库的max\_connections必须要大于主库的

然后再从库，

1先形成一个主库的基本备份

2、配置从机 recover.conf ,开启流复制

standby\_mode = on  #指明从库身份

primary\_conninfo = 'host=10.0.120.150 port=5432 user=repl password=repl1234'  #连接到主库信息

recovery\_target\_timeline = 'latest'     #同步到最新数据

trigger\_file = ‘/opt/postgres/9.6/data/trigger\_activest’指定主机切换触发文件

3.配置从机postgresql.conf文件

hot\_standby=on //允许复制时查询

max\_connections = xxx, # 一般比主库大一些.

wal\_receiver\_status\_interval = 10s

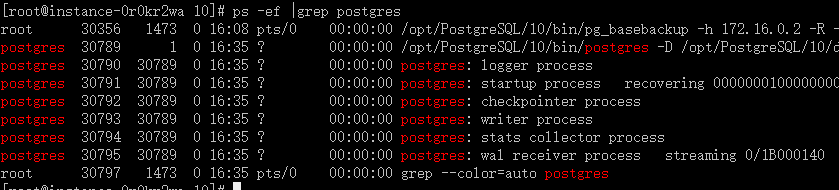
hot\_standby\_feedback = on # 如果有错误的数据复制,是否向 主进行范例.

3、重新启动

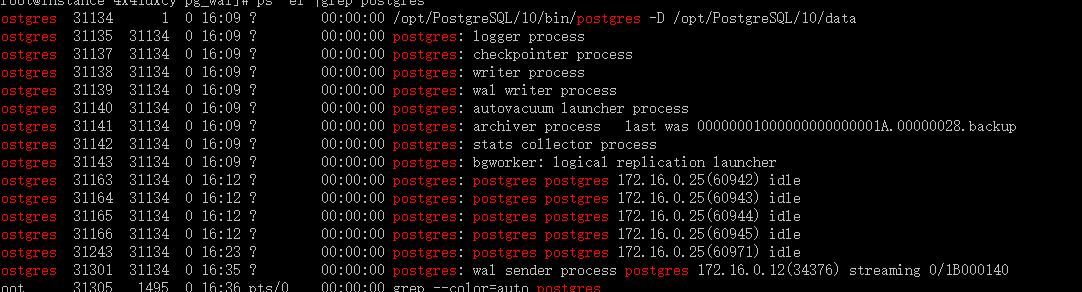
1 和2 可以通过 pg\_basebackup  -R -h -U -D 一步到位

/opt/PostgreSQL/9.6/bin/pg\_basebackup -h 106.13.55.240 -R -D /opt/PostgreSQL/9.6/data -U postgres -W

从库下的进程



主库下的进程



二： 归档恢复  archive recover

一个wal日志 一个wal日志的恢复

关键在看懂 recover.conf   pg\_home/share/下有例子

# When standby\_mode is enabled, the PostgreSQL server will work as a

# standby. It will continuously wait for the additional XLOG records, using

# restore\_command and/or primary\_conninfo.

standby\_mode=on 的时候，从数据库不允许其他读写，只允许自己内部的 restore ；

流复制的时候，使用的是 primary\_conninfo 自动复制、恢复

日志复制，需要执行 restore\_command 命令 ，来达到同样的效果；

PITR恢复是基于文件系统备份和wal文件的备份

若standby\_mode=off ,可手动启动恢复，by 重启服务

若 standby\_mode=on  , 自动执行restore\_command

A:先看如何手动通过 基础备份+wal日志 来恢复一个数据库的

1 通过 pg\_basebackup 获得了一  data文件夹；

2 在另外一台机器上有 pg的安装文件（不包括data目录或删除掉）

3 将新的wal日志复制到待恢复的机器上

4 启动服务

可在recover.conf 中设置 一些选项，控制恢复行为；比如  recovery\_target\_time = '2016-04-21 14:49:14' 控制恢复的截止时间

wal复制，完整操作流程  （某个方案）

1 在从库上通过pg\_basebackup 获得基础备份

2 在主库中定义 archive\_mode= on

archive\_command='scp %p postgres@172.16.0.12:/opt/PostgreSQL/data/pg\_wal/%f'

每当归档条件触发，则执行archive\_command ,以postgres身份执行

给予postgres sudo条件，遇到问题

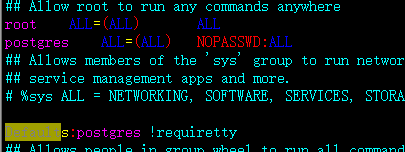
sudo: 没有终端存在，且未指定 askpass 程序

解决方法

在 /etc/sudoers 里添加

postgres ALL =(ALL)  NOPASSWD:ALL

Defaults :postgres !requiretty



3 在从库设置 recover.conf

standby\_mode=on

resotre\_command =''

standby\_mode (boolean)指定是否将PostgreSQL服务器作为一个后备服务器启动。

如果这个参数为on，当到达已归档 WAL 末尾时该服务器将不会停止恢复， 但是将通过使用restore\_command获得 新的 WAL 段以及/或者通过使用 primary\_conninfo设置连接到主服务器来尝试继续恢复

如果standby\_mode为off，设置 primary\_conninfo没有效果。

2 和3 的目的就是将新产生的wal文件,复制到 从库的pg\_wal下，同时保证postgres权限

archive\_command='scp %p postgres@172.16.0.12:/opt/PostgreSQL/data/pg\_wal/%f'

resotre\_command =''

这样的组合，问题很大；比如设置两个postgres用户之间的免密，或者命令行密码问题；

需要 两个postgres 之间免密， 为 /opt/PostgreSQL/10 的owner设置为postgres;

scp pg\_wal/00000001000000000000004A postgres@172.16.0.12:/opt/PostgreSQL/10/data/pg\_wal/00000001000000000000004A

注意 文件夹权限问题 chmod 755 /path

显然这不是最佳方案；NFS中间共享磁盘  +

archive\_command='cp %p /share/path1/%f'

resotre\_command ='cp /share/path1/%f %p'

更好。

方案可行二：

1 在从库上通过pg\_basebackup 获得基础备份

2 在主库中定义 archive\_mode= on

archive\_command='cp %p /data/%f'

chown postgres:postgres /data

3 从库上编辑 recovery.conf

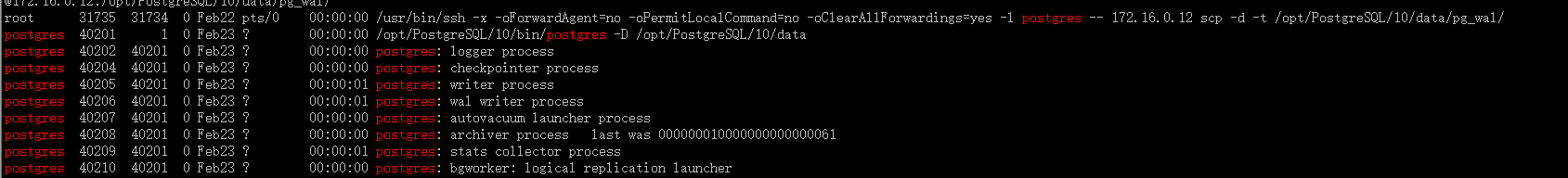
standby\_mode=on

restore\_command ='scp postgres@172.16.0.'

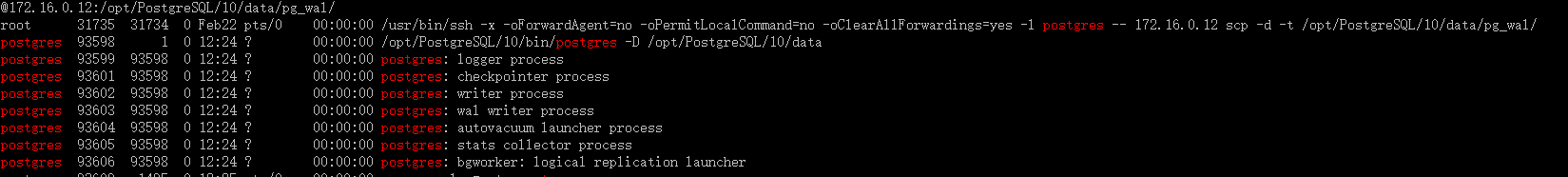
standby\_mode下的进程



物理复制下的进程



逻辑复制下修改 wal\_level=logical 同时 注释掉 archive\_mode  和 archive\_command后的进程



主从切换

查看主从

Select pg\_is\_in\_recovery();

主机为f,从机为t

ps -ef | grep wal

wal 日志发送进程"wal sender process"，说明是主库。  
wal 日志接收进程"wal receiver process" ，说明是备库；

切换

前提:

recovery.conf 需做如下配置:

standby\_mode = on  #指明从库身份

primary\_conninfo = 'host=10.0.120.150 port=5432 user=repl password=repl1234'  #连接到主库信息

recovery\_target\_timeline = 'latest'     #同步到最新数据

trigger\_file = ‘/opt/postgres/9.6/data/trigger\_activest’

指定触发文件，文件存在时，将触发从库提升为主库，前提是必须设置”standby\_mode = on”；如果不设置此参数，也可采用”pg\_ctl promote“触发从库切换成主库

主机宕机后,只需touch trigger\_file 指定文件, recovery.conf会变成recovery.done,从机就会变成主机

同步异步

数据同步方式时: 在master 提交事务时, 一定会等到WAL 日志传递到standby 后才返回, 这样可以得到standby 数据完全和 master 数据库同步. 没有一点落后.(自动切换,可以达到零丢失)

数据异步同步方式: master 提交事务后,不必等日志传递到standby 就即可返回. 所以standby 数据库通常 比master 落后很少.

逻辑复制

支持Postgresql>=10版本

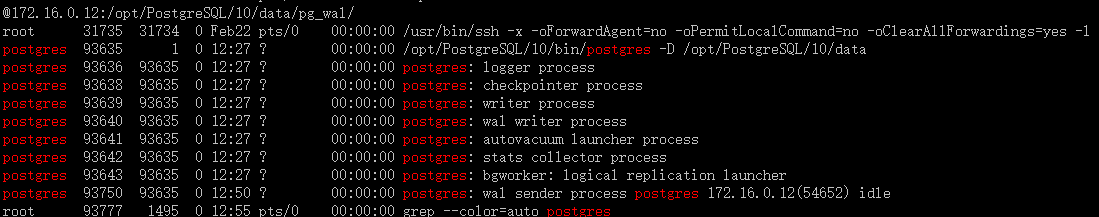
在master某张表上创建 发布者

create publication  test10\_pub for table test10;

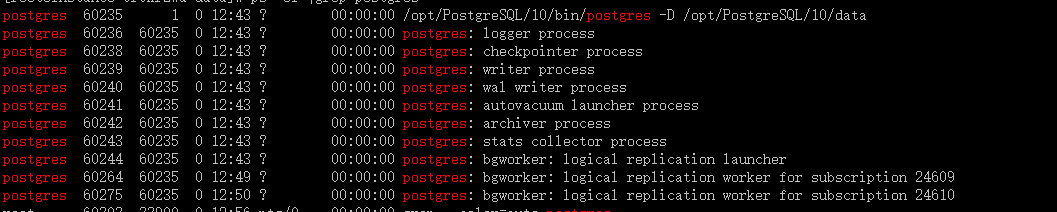
在 subscr 上 创建 订阅

create subscription test10\_sub connection 'host=172.16.0.10 user=postgres password=since2015 dbname=postgres' publication test10\_pub;

此时 在master上 的进程



subscr上的进程



注意事项 和概念

master 需要设置wal\_level=logical，同时开启足够的worker，设置足够大的replication slot，设置足够多的sender。

因为每一个订阅，都要消耗掉一个replication slot，需要消耗一个wal sender，一个worker进程。

subscr，需要指定master的连接信息，以及 subscription name，同时指定需要在publication数据库中创建的publication name。

在同一个数据库中，可以创建多个subscr。

subscr和master的角色可以同时出现在同一个实例的同一个数据库中。

注意，subscr一样需要设置：

max\_replication\_slots

max\_logical\_replication\_workers

max\_worker\_processe

master

1. wal\_level=logical

2. max\_replication\_slots，每一个订阅需要消耗一个slot，每一个指定了slot的流式物理复制也要消耗一个slot。

3. max\_wal\_senders，每一个slot要使用一个wal sender，每一个流式物理复制也要使用一个wal sender。

4. max\_worker\_processes，必须大于等于max\_wal\_senders加并行计算进程，或者其他插件需要fork的进程数。

Subscr

1. max\_replication\_slots，大于等于该实例总共需要创建的订阅数

2. max\_logical\_replication\_workers，大于等于该实例总共需要创建的订阅数

3. max\_worker\_processes， 大于等于max\_logical\_replication\_workers + 1 + CPU并行计算 + 其他插件需要fork的进程数.

逻辑复制，本质上是事务层级的复制，需要在订阅端执行SQL。

如果订阅端执行SQL失败（或者说引发了任何错误，包括约束等），都会导致该订阅暂停。

注意，update, delete没有匹配的记录时，不会报错，也不会导致订阅暂停。

用户可以在订阅端数据库日志中查看错误原因。

逻辑复制，是跨平台的 ，跨版本的