## 集群配置

节点内存：2\*500G

CPU（ARM）： 4\*24 ,

磁盘：1.7TB \*2，2块盘做raid0

系统：Centos7-minial

集群方式：1主1备

## 连接配置

配置公式：

max\_pool，num\_init\_children，max\_connections和superuser\_reserved\_connections 必须符合以下规则：

max\_pool\*num\_init\_children <= (max\_connections - superuser\_reserved\_connections) (不需要取消查询)  
max\_pool\*num\_init\_children\*2 <= (max\_connections - superuser\_reserved\_connections) (需要取消查询)

现场测试配置：

max\_pool=2

num\_init\_children=12000

max\_connections=25000

superuser\_reserved\_connections=3

其他参数：

shared\_buffers = 30GB

max\_wal\_size = 15GB

写入方式：copy单个大文件

12GB，耗时214s

写入速度：57MB/s

每分钟新增订单数tpmC：19w左右

## 全量备份与还原

准备数据：

create database test;

create database mydb;

数据库开启归档（postgresql.conf增加）：

1. 创建归档目录

mkdir /usr/txdbdata/archived\_wals

1. 数据库开启归档

archive\_mode = on

archive\_command = 'cp %p /usr/txdbdata/archived\_wals/%f'

1. 重启数据库

pg\_ctl restart

创建备份目录

export BACKUP\_PATH=/home/c/pg\_rman\_bk

1. 初始化rman

pg\_rman init

1. 全量备份
2. pg\_rman backup --backup-mode=full -B /home/txadmin/pg\_rman\_bk/ -C -P

验证

pg\_rman validate

查看备份

pg\_rman show

1. 删除数据库
2. Drop database testdb
3. 数据库恢复

停止要恢复的数据库

pg\_ctl stop

pg\_rman restore --recovery-target-time="2021-04-08 22:02:17"（指定备份的结束时间）

启动数据库

pg\_ctl start

重放redolog(日志里有提示的话，再操作，否则不需要)

select pg\_wal\_replay\_resume();

查看恢复是否ok

select datname from pg\_database;

## 增量备份与还原

支持对指定库进行增量备份和还原功能。

数据库开启归档：

archive\_mode = on

archive\_command = 'cp %p /usr/txdbdata/archived\_wals/%f'

设置备份目录：

export BACKUP\_PATH=/home/txadmin/pg\_rman\_bk

重启数据库：

pg\_ctl restart

初始化pg\_rman:

pg\_rman init

全库备份：

pg\_rman backup --backup-mode=full -B /home/txadmin/pg\_rman\_bk/ -C -P

验证备份：

pg\_rman validate

查看备份：

pg\_rman show

测试数据准备

create table t2(a int);

insert into t2 values (generate\_series(1,1000000));

create table t3(a int);

insert into t3 values (generate\_series(1,1000000));

数据库进行增量备份

pg\_rman backup --backup-mode=incremental

验证备份：

pg\_rman validate

查看备份

pg\_rman show

删除表t2、t3

drop table t2;

drop table t3;

停止要还原的数据库：

pg\_ctl stop

执行还原操作：

pg\_rman restore --recovery-target-time="2021-04-08 22:22:35"(指定备份的结束时间)

启动数据库：

pg\_ctl start

重放redolog(日志里有提示的话，再操作，否则不需要)：

select pg\_wal\_replay\_resume();

验证恢复是否成功：

\d

关联列表

架构模式 | 名称 | 类型 | 拥有者

----------+------+--------+----------

public | t2 | 数据表 | stxadmin

public | t3 | 数据表 | stxadmin

(2 行记录)

## 主备切换

1.pgpool配置如下

failover\_command = '/usr/tx\_pgpool/etc/failover.sh %d %h %p %D %m %H %M %P %r %R %M %S'

2.打开pgpool日志

tailf /var/log/pgpool/pgpool.log

3.停止数据库主节点

pg\_ctl stop

4.pgpool日志输出

2021-04-09 15:14:00: pid 46629: LOG: execute command: /usr/tx\_pgpool/etc/failover.sh 0 node2 6868 /usr/txdbdata 1 node3 0 0 6868 /usr/txdbdata 0 6868

+ FAILED\_NODE\_ID=0

+ FAILED\_NODE\_HOST=node2

+ FAILED\_NODE\_PORT=6868

+ FAILED\_NODE\_PGDATA=/usr/txdbdata

+ NEW\_MAIN\_NODE\_ID=1

+ NEW\_MAIN\_NODE\_HOST=node3

+ OLD\_MAIN\_NODE\_ID=0

+ OLD\_PRIMARY\_NODE\_ID=0

+ NEW\_MAIN\_NODE\_PORT=6868

+ NEW\_MAIN\_NODE\_PGDATA=/usr/txdbdata

+ OLD\_PRIMARY\_NODE\_HOST=0

+ OLD\_PRIMARY\_NODE\_PORT=6868

+ PGHOME=/usr/txdb4.0

+ echo failover.sh: start: failed\_node\_id=0 old\_primary\_node\_id=0 failed\_host=node2 new\_main\_host=node3

failover.sh: start: failed\_node\_id=0 old\_primary\_node\_id=0 failed\_host=node2 new\_main\_host=node3

+ '[' 1 -lt 0 ']'

+ ssh -T txadmin@node3 ls /tmp

+ '[' 0 -ne 0 ']'

+ '[' 0 -ne 0 ']'

+ echo failover.sh: Primary node is down, promote standby node node3.

failover.sh: Primary node is down, promote standby node node3.

+ ssh -T txadmin@node3 /usr/txdb4.0/bin/pg\_ctl -D /usr/txdbdata -w promote

等待服务器进程加载 .... 完成

服务器加载完毕

+ '[' 0 -ne 0 ']'

+ echo failover.sh: end: new\_main\_node\_id=1 is promoted to a primary

failover.sh: end: new\_main\_node\_id=1 is promoted to a primary

+ exit 0

2021-04-09 15:14:01: pid 46629: LOG: find\_primary\_node\_repeatedly: waiting for finding a primary node

2021-04-09 15:14:01: pid 46629: LOG: find\_primary\_node: primary node is 1

2021-04-09 15:14:01: pid 46629: LOG: failover: set new primary node: 1

2021-04-09 15:14:01: pid 46629: LOG: failover: set new main node: 1

2021-04-09 15:14:02: pid 58661: LOG: worker process received restart request

2021-04-09 15:14:02: pid 46631: LOG: new IPC connection received

2021-04-09 15:14:02: pid 46631: LOG: received the failover indication from Pgpool-II on IPC interface

2021-04-09 15:14:02: pid 46631: LOG: watchdog is informed of failover end by the main process

failover done. shutdown host node2(6868)2021-04-09 15:14:02: pid 46629: LOG: failover done. shutdown host node2(6868)

2021-04-09 15:14:03: pid 58660: LOG: restart request received in pcp child process

2021-04-09 15:14:03: pid 46629: LOG: PCP child 58660 exits with status 0 in failover()

5.连接到备机

[txadmin@node3 txdbdata]$ ps -ef|grep postgres

txadmin 19733 1 0 15:19 ? 00:00:00 /usr/txdb4.0/bin/postgres

txadmin 19734 19733 0 15:19 ? 00:00:00 postgres: logger

txadmin 19737 19733 0 15:19 ? 00:00:00 postgres: checkpointer

txadmin 19738 19733 0 15:19 ? 00:00:00 postgres: background writer

txadmin 19739 19733 0 15:19 ? 00:00:00 postgres: stats collector

txadmin 20004 19733 0 15:23 ? 00:00:00 postgres: walwriter

txadmin 20005 19733 0 15:23 ? 00:00:00 postgres: autovacuum launcher

txadmin 20006 19733 0 15:23 ? 00:00:00 postgres: archiver last was 000000010000000200000093.partial

txadmin 20007 19733 0 15:23 ? 00:00:00 postgres: logical replication launcher

txadmin 20029 19733 0 15:23 ? 00:00:00 postgres: txadmin testdb 10.220.50.52(52476) idle

txadmin 20031 19563 0 15:23

确认已经成功激活

## 扩容/缩容

在线恢复（注意：数据库主节点的postgresql.auto.conf 必须为空）

免密配置

txadmin(node2)

ssh-keygen

ssh-copy-id -i ~/.ssh/id\_rsa.pub txadmin@node3

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@node3

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@node2

ssh-copy-id -i ~/.ssh/id\_rsa.pub txadmin@node2

root(node2)

ssh-keygen

ssh-copy-id -i ~/.ssh/id\_rsa.pub txadmin@node3

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@node3

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@node2

ssh-copy-id -i ~/.ssh/id\_rsa.pub txadmin@node2

pgpool配置

recovery\_1st\_stage\_command = 'recovery\_1st\_stage'

新增backend配置：

backend\_hostname1 = ‘xx’

backend\_port1 = 6868

backend\_weight1 = 1

backend\_data\_directory1 = '/usr/txdbdata'

backend\_flag1 = 'ALLOW\_TO\_FAILOVER'

backend\_application\_name1 = 'server1'

1.pgdata目录新增文件，主备

pgpool\_remote\_start、recovery\_1st\_stage

授权

chmod u+x recovery\_1st\_stage

chmod u+x pgpool\_remote\_start

2.创建扩展

psql template1 -c "CREATE EXTENSION pgpool\_recovery"

3.执行扩容

pcp\_recovery\_node -h node2 -p 9898 -n 0

注意：-h指定pgpool所在主机 –n指定新的节点id，从当前最大值推算+1。

pcp\_recovery\_node 实际上会执行Pg\_basebackup操作和remote\_start操作

4.查看pg主节点日志

+ PRIMARY\_NODE\_PGDATA=/usr/txdbdata

+ DEST\_NODE\_HOST=node3

+ DEST\_NODE\_PGDATA=/usr/txdbdata

+ PRIMARY\_NODE\_PORT=6868

+ DEST\_NODE\_ID=1

+ DEST\_NODE\_PORT=6868

++ hostname

+ PRIMARY\_NODE\_HOST=node2

+ PGHOME=/usr/txdb4.0

+ ARCHIVEDIR=/usr/txdbdata/archived\_wals

+ REPLUSER=txadmin

+ echo recovery\_1st\_stage: start: pg\_basebackup for Standby node 1

recovery\_1st\_stage: start: pg\_basebackup for Standby node 1

+ ssh -T -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null txadmin@node3 -i /home/txadmin/.ssh/id\_rsa ls /tmp

Warning: Permanently added 'node3,10.220.50.53' (ECDSA) to the list of known hosts.^M

+ '[' 0 -ne 0 ']'

++ /usr/txdb4.0/bin/initdb -V

++ awk '{print $3}'

++ sed 's/\..\*//'

++ sed 's/\([0-9]\*\)[a-zA-Z].\*/\1/'

+ PGVERSION=12

+ '[' 12 -ge 12 ']'

+ RECOVERYCONF=/usr/txdbdata/myrecovery.conf

+ /usr/txdb4.0/bin/psql -p 6868

pg\_create\_physical\_replication\_slot

-------------------------------------

(node3,)

(1 行记录)

++ echo /usr/txdbdata/myrecovery.conf

++ sed -e 's/\//\\\//g'

++ echo /usr/txdbdata/myrecovery.conf

++ sed -e 's/\//\\\//g'

+ ssh -T -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null txadmin@node3 -i /home/txadmin/.ssh/id\_rsa '

set -o errexit

rm -rf /usr/txdbdata/\*

rm -rf /usr/txdbdata/archived\_wals/\*

source /home/txadmin/.bash\_profile

/usr/txdb4.0/bin/pg\_basebackup -h node2 -U txadmin -p 6868 -D /usr/txdbdata -X stream

if [ 12 -ge 12 ]; then

sed -i -e "\$ainclude\_if\_exists = '\''\/usr\/txdbdata\/myrecovery.conf'\''" -e "/^include\_if\_exists = '\''\/usr\/txdbdata\/myrecovery.conf'\''/d" /usr/txdbdata/postgresql.conf

fi

cat > /usr/txdbdata/myrecovery.conf << EOT

primary\_conninfo = '\''host=node2 port=6868 user=txadmin application\_name=node3 passfile='\'''\''/home/txadmin/.pgpass'\'''\'''\''

recovery\_target\_timeline = '\''latest'\''

restore\_command = '\''scp node2:/usr/txdbdata/archived\_wals/%f %p'\''

primary\_slot\_name = '\''node3'\''

EOT

if [ 12 -ge 12 ]; then

touch /usr/txdbdata/standby.signal

else

echo "standby\_mode = '\''on'\''" >> /usr/txdbdata/myrecovery.conf

fi

sed -i "s/#\*port = .\*/port = 6868/" /usr/txdbdata/postgresql.conf

'

Warning: Permanently added 'node3,10.220.50.53' (ECDSA) to the list of known hosts.^M

+ '[' 0 -ne 0 ']'

+ echo recovery\_1st\_stage: end: recovery\_1st\_stage is completed successfully

recovery\_1st\_stage: end: recovery\_1st\_stage is completed successfully

+ exit 0

+ DEST\_NODE\_HOST=node3

+ DEST\_NODE\_PGDATA=/usr/txdbdata

+ PGHOME=/usr/txdb4.0

+ echo pgpool\_remote\_start: start: remote start Standby node node3

pgpool\_remote\_start: start: remote start Standby node node3

+ ssh -T -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null txadmin@node3 -i /home/txadmin/.ssh/id\_rsa ls /tmp

Warning: Permanently added 'node3,10.220.50.53' (ECDSA) to the list of known hosts.^M

+ '[' 0 -ne 0 ']'

+ ssh -T -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null txadmin@node3 -i /home/txadmin/.ssh/id\_rsa '

/usr/txdb4.0/bin/pg\_ctl -l /dev/null -w -D /usr/txdbdata start

'

Warning: Permanently added 'node3,10.220.50.53' (ECDSA) to the list of known hosts.^M

等待服务器进程启动 .... 完成

服务器进程已经启动

+ '[' 0 -ne 0 ']'

+ echo pgpool\_remote\_start: end: PostgreSQL on node3 is started successfully.

pgpool\_remote\_start: end: PostgreSQL on node3 is started successfully.

+ exit 0

5.连接恢复的节点

[txadmin@node3 txdbdata]$ ps -ef|grep postgres

txadmin 19132 1 0 10:34 ? 00:00:00 /usr/txdb4.0/bin/postgres -D /usr/txdbdata

txadmin 19133 19132 0 10:34 ? 00:00:00 postgres: logger

txadmin 19134 19132 0 10:34 ? 00:00:00 postgres: startup recovering 00000005000000000000002C

txadmin 19142 19132 0 10:35 ? 00:00:00 postgres: checkpointer

txadmin 19143 19132 0 10:35 ? 00:00:00 postgres: background writer

txadmin 19145 19132 0 10:35 ? 00:00:00 postgres: stats collector

txadmin 19149 19132 0 10:35 ? 00:00:00 postgres: walreceiver streaming 0/2C000148

txadmin 19170 13311 0 10:35 pts/1 00:00:00 tailf postgresql-Mon.log

txadmin 19254 19132 0 10:36 ? 00:00:00 postgres: txadmin testdb 10.220.50.52(47334) idle

txadmin 19317 19132 0 10:37 ? 00:00:00 postgres: txadmin txdb\_default 10.220.50.52(47394) idle

txadmin 19435 19132 0 10:39 ? 00:00:00 postgres: txadmin testdb 10.220.50.52(47492) idle

txadmin 19609 13368 0 10:42 pts/4 00:00:00 grep --color=auto postgres

6.查看流复制视图

txdb\_default=# \x

扩展显示已打开.

txdb\_default=# select \*from pg\_stat\_replication;

-[ RECORD 1 ]----+------------------------------

pid | 92398

usesysid | 16384

usename | replicator

application\_name | walreceiver

client\_addr | 10.220.50.53

client\_hostname |

client\_port | 54798

backend\_start | 2021-04-12 10:25:56.261082+08

backend\_xmin |

state | streaming

sent\_lsn | 0/2C000148

write\_lsn | 0/2C000148

flush\_lsn | 0/2C000148

replay\_lsn | 0/2C000148

write\_lag |

flush\_lag |

replay\_lag |

sync\_priority | 0

sync\_state | async

reply\_time | 2021-04-12 10:44:10.263556+08

## 负载均衡

开启负载均衡:

pgpool.conf

load\_balance\_mode = on

实测是会话级的负载均衡，每一个连接从连接上pgpool之后，负载均衡就确定了，执行语句就会一直分发到同一个节点。可以确定，负载均衡在连接建立时就已确定。

启动测试程序

nohup ./runBenchmark.sh props.txdb &

查看结果

show pool\_backend\_stats;

node\_id | hostname | port | status | role | select\_cnt | insert\_cnt | update\_cnt | delete\_cnt | ddl\_cnt | other\_cnt | panic\_cnt | fatal\_cnt | error\_cnt

---------+----------+------+--------+---------+------------+------------+------------+------------+---------+-----------+-----------+-----------+-----------

0 | node0 | 6868 | up | primary | 627246 | 286778 | 363879 | 19736 | 36 | 97521 | 0 | 0 | 36

1 | node4 | 6868 | up | standby | 4657 | 0 | 0 | 0 | 0 | 48984 | 0 | 0 | 3

(2 行记录)

备注:

数据库集群内部会针对SELECT语句按照内部的规则分发到不同的节点来分担负载，show pool\_backend\_stats会显示提交到不同数据库节点的select、update、insert等语句个数，如果 select\_cnt在standby上不为0，则说明读写分离正常。

## 性能测试方案

1：在数据仓数固定的情况下增加并发数，寻找tpmC(new order)最大时的并发数，记录结果

2：调大数据仓数，依次增加并发数，寻找此仓数下的tpmC(new order)最大时的并发数

继续增加数据仓数，直到不再tmpC继续增加

可以认为，tmpC最大时的数据仓数、并发数为在此种情况下的机器最大性能。