**pg集群搭建-20190607**

**一，方案分析**

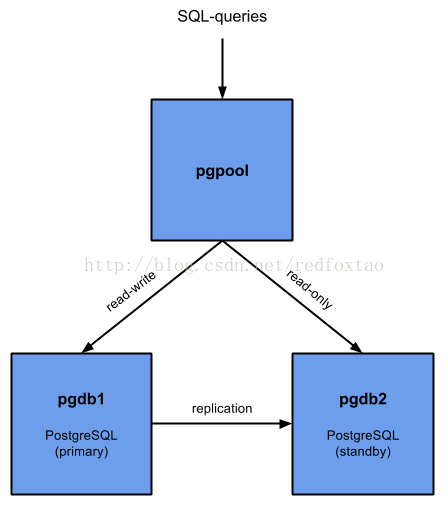
最近研究了PG的两种集群方案，分别是Pgpool-II和Postgres-XL，在这里总结一下二者的机制、结构、优劣、测试结果等。

1、 Pgpool-II和Postgres-XL简介

据我目前的了解，Pgpool-II和Postgres-XL是PG集群开源实现中比较成功的两个项目，其中Pgpool-II的前身的Pgpool-I，Postgres-XL的前身是Postgres-XC。

1.1、Pgpool-II

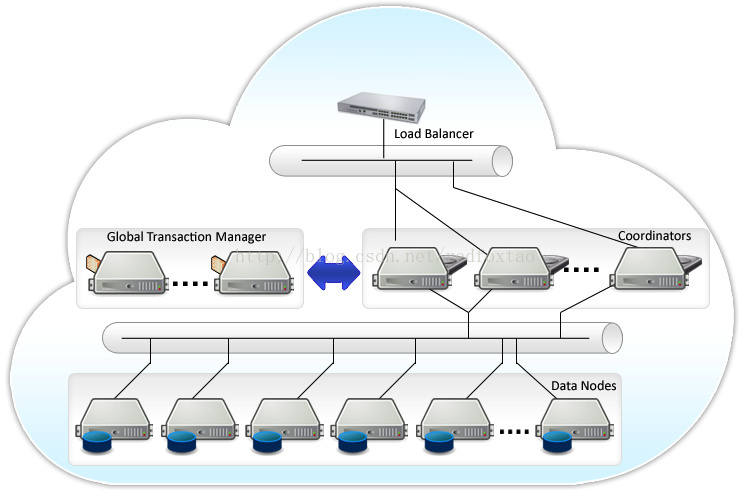
Pgpool-II相当于中间件，位于应用程序和PG服务端之间，对应用程序来说，Pgpool-II就相当于PG服务端；对PG服务端来说，Pgpool-II相当于PG客户端。由此可见，Pgpool-II与PG是解耦合的，基于这样的机制，Pgpool-II可以搭建在已经存在的任意版本的PG主从结构上，主从结构的实现与Pgpool-II无关，可以通过slony等工具或者PG自身的流复制机制实现。除了主从结构的集群，Pgpool-II也支持多主结构，称为复制模式，该模式下PG节点之间是对等的，没有主从关系，写操作同时在所有节点上执行，这种模式下写操作的代价很大，性能上不及主从模式。PG 9.3之后支持的流复制机制可以方便的搭建主从结构的集群（包括同步复制与异步复制），因此Pgpool-II中比较常用的模式是流复制主从模式（也可以一主多从）。



既然PG可以通过自身的流复制机制方便的搭建主从结构集群，为什么还要在它上面搭建Pgpool-II呢？因为简单的主从结构集群并不能提供连接池、负载均衡、自动故障切换等功能，Pgpool-II正好可以做到这些，当然负载均衡只针对读操作，写操作只发生在主节点上。为了避免单点故障，Pgpool-II自身也可以配置为主从结构，对外提供虚拟IP地址，当主节点故障后，从节点提升为新的主节点并接管虚拟IP。

1.2、Postgres-XL

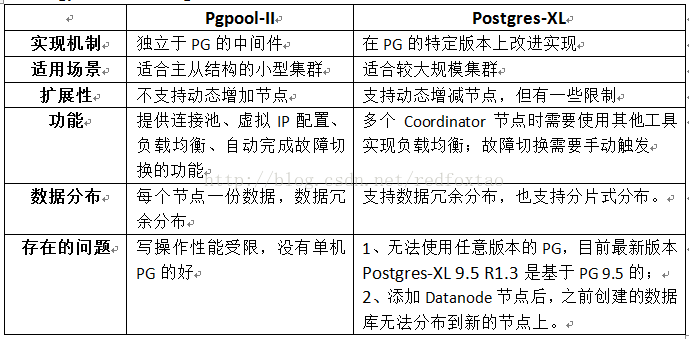
Postgres-XL的机制和Pgpool-II大不相同，它不是独立于PG的，是在PG源代码的基础上增加新功能实现的。简单来说，Postgres-XL将PG的SQL解析层的工作和数据存取层的工作分离到不同的两种节点上，分别称为Coordinator节点和Datanode节点，而且每种节点可以配置多个，共同协调完成原本单个PG实例完成的工作。此外，为了保证分布模式下事务能够正确执行，增加了一个GTM节点。为了避免单点故障，可以为所有节点配置对应的slave节点。Postgres-XL结构图见下图，来自官网。



Postgres-XL的Coordinator节点是整个集群的数据访问入口，可以配置多个，然后在它们之上通过Nginx等工具实现负载均衡。Coordinator节点维护着数据的存储信息，但不存储数据本身。接收到一条SQL语句后，Coordinator解析SQL，制定执行计划，然后分发任务到相关的Datanode上，Datanode返回执行结果到Coordinator，Coordinator整合各个Datanode返回的结果，最后返回给客户端。

Postgres-XL的Datanode节点负责实际存取数据，数据在多个Datanode上的分布有两种方式：复制模式和分片模式，复制模式下，一个表的数据在指定的节点上存在多个副本；分片模式下，一个表的数据按照指定的规则分布在多个数据节点上，这些节点共同保存一份完整的数据。这两种模式的选择是在创建表的时候执行CREATE TABLE语句指定的，也可以通过ALTER TABLE语句改变数据的分布方式。

2、 Pgpool-II和Postgres-XL对比



3、 Pgpool-II和Postgres-XL的性能测试

我分别使用pgbench和benchmarksql测试了Pgpool-II集群和Postgres-XL集群的性能，为了对比，还测试单机PG的性能。

测试条件：Pgpool-II集群是搭建在两台虚机上的主从复制（异步）集群；Postgres-XL集群也是搭建在相同条件上的两台虚机的集群，其中包含两个Coordinator节点和两个Datanode节点。单机PG也是运行在相同条件的虚机上。操作系统是CentOS 6.6，单机PG和Pgpool-II集群种的PG版本号是9.5，Postgres-XL的版本号是Postgres-XL 9.5 R1.3，也只基于PG 9.5的。

3.1、pgbench测试

pgbench是PG自带的一款简单的PG性能测试工具，测试指标是TPS，表示每秒钟完成的事务数。测试过程如下：

1) 建库

psql -h 10.192.33.244 -p7777 -c "create database pgbench"

2) 生成数据

pgbench -i -s 1000 -h 10.192.33.244 -p 7777 pgbench

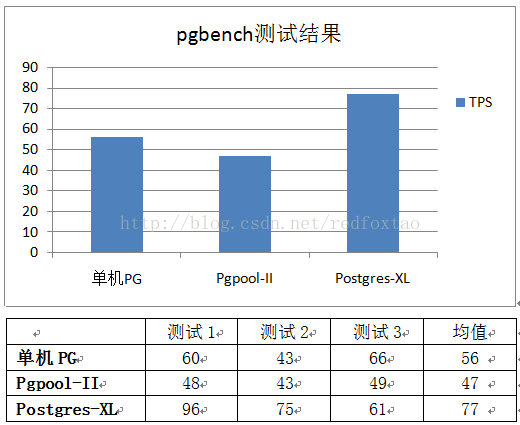
#参数-s指定数据量，这里使用1000，最终生成的数据量大小约16G。

3) 测试

pgbench -h 10.192.33.244 -p7777 -c30 -T300 -n

#测试时间5分钟，连续测试3次。

pgbench测试结果：



pgbench的测试结果显示，Pgpool-II集群的性能比单机PG的性能差一些，约为84%；Postgres-XL集群的性能比单机PG的性能好一些，约为137%。

3.2、benchmarksql测试

benchmarksql的是一款常用的TPC-C测试工具，TPC-C测试衡量的是数据库的OLTP性能。测试过程如下：

1) 建库

psql -h 10.192.33.244 -p7777 -c "create database tpcc"

2) 生成数据

./runDatabaseBuild.sh props.pg

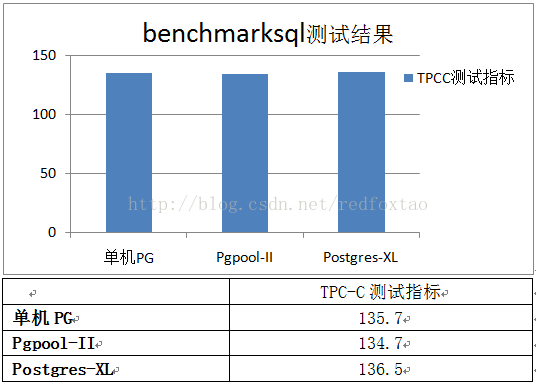
***#props****.pg为配置文件，配置数据库链接信息以及测试数据量、测试时间等，*

*#这里配置的数据量是100 warehouse，最终生成的数据约10G，测试时间1小时。*

3) 测试

./runBenchmark.sh props.pg

benchmarksql测试结果：



benchmarksql测试结果显示，两种集群与单机PG的性能指标几乎一致，无法分辨高下。出现这种结果的可能原因之一是：测试数据量较小，无法发挥集群的性能优势，尤其像Postgres-XL这个集群在设计上针对大数据处理做了一些优化，应该更加适合大数据处理的场景。鉴于benchmarksql测试生成数据十分耗时，这里就不再进行较大数据量的测试了。

最后，综合来看，我更倾向于Postgres-XL，如果公司今后打算用的话，我会推介。

**二，准备环境搭建**

1搭建linux环境

选取centos7做系统

root：123456

1.1永久修改主机名：hostnamectl set-hostname postgres1/2/3

重启系统reboot

查看hostname： cat /etc/hostname 或者 hostname

1.2设置网卡：ifconfig 查看虚拟机网关名称ens160 发现没有ip地址

cat /etc/sysconfig/network-scripts/ifcfg-ens160修改网络配置

TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4\_FAILURE\_FATAL=no
IPV6INIT=yes
IPV6\_AUTOCONF=yes
IPV6\_DEFROUTE=yes
IPV6\_PEERDNS=yes
IPV6\_PEERROUTES=yes
IPV6\_FAILURE\_FATAL=no
IPV6\_ADDR\_GEN\_MODE=stable-privacy
NAME=ens32
UUID=29c76cc1-6688-404f-8537-81a3561c3bf9
DEVICE=ens32
ONBOOT=yes
IPADDR=192.168.2.130
NETMASK=255.255.255.0
GATEWAY=192.168.2.1
DNS1=210.22.70.3
DNS2=8.8.8.8

/etc/init.d/network restart

ping 内网外网地址 正常 用xshell连接查看正常

os: centos 7

pgxl:pg.version '10.3 (Postgres-XL 10alpha2)

pgxl 是一款非常实用的横向扩展的开源软件，继承了很多pgxc的功能，在replication 和sharding 方面有着非常棒的用处。

pgxl 不严格的说是 pgxc的升级加强版。是对官方 postgresql 的版本的修改提升，为大牛点赞。

Global Transaction Monitor (GTM)

全局事务管理器，确保群集范围内的事务一致性。 GTM负责发放事务ID和快照作为其多版本并发控制的一部分。

集群可选地配置一个备用GTM，以改进可用性。此外，可以在协调器间配置代理GTM， 可用于改善可扩展性，减少GTM的通信量。

GTM Standby

GTM的备节点，在pgxc,pgxl中，GTM控制所有的全局事务分配，如果出现问题，就会导致整个集群不可用，为了增加可用性，增加该备用节点。当GTM出现问题时，GTM Standby可以升级为GTM，保证集群正常工作。

GTM-Proxy

GTM需要与所有的Coordinators通信，为了降低压力，可以在每个Coordinator机器上部署一个GTM-Proxy。

Coordinator

协调员管理用户会话，并与GTM和数据节点进行交互。协调员解析，并计划查询，并给语句中的每一个组件发送下一个序列化的全局性计划。

为节省机器，通常此服务和数据节点部署在一起。

Data Node

数据节点是数据实际存储的地方。数据的分布可以由DBA来配置。为了提高可用性，可以配置数据节点的热备以便进行故障转移准备。

总结：

gtm是负责ACID的，保证分布式数据库全局事务一致性。得益于此，就算数据节点是分布的，但是你在主节点操作增删改查事务时，就如同只操作一个数据库一样简单。

Coordinator是调度的，将操作指令发送到各个数据节点。

datanodes是数据节点，分布式存储数据。

规划如下：

node1 192.168.2.164 gtm

node2 192.168.2.165 gtm-proxy,coordinator,datanode

node3 192.168.2.166 gtm-proxy,coordinator,datanode

下载

Postgres-XL 9.5 R1.6发布 - 2017年8月24日

Postgres-XL 10 R1.1 发布2019年2月18日

官方瞎咋i地址<https://www.postgres-xl.org/>

https://git.postgresql.org/gitweb/?p=postgres-xl.git;a=summary

git://git.postgresql.org/git/postgres-xl.git

**三.安装集群环境**

以下操作除了 mkdir {gtm,gtm\_slave,pgxc\_ctl} 命令外都需要在node2、node3 上执行

node1 需要安装依赖包

# yum install -y bison flex perl-ExtUtils-Embed readline-devel zlib-devel pam-devel libxml2-devel libxslt-devel openldap-devel python-devel gcc gcc-c++ openssl-devel cmake openjade docbook-style-dsssl uuid uuid-devel

node1 节点上关闭防火墙，selinux

# systemctl stop firewalld.service
# systemctl disable firewalld.service
# vim /etc/selinux/config
disabled

node1 节点上创建用户

# groupadd postgres
# useradd postgres -g postgres
# passwd postgres
# mkdir -p /usr/pgxl-10
# chown -R postgres:postgres /usr/pgxl-10
# mkdir -p /var/lib/pgxl
# cd /var/lib/pgxl
# mkdir {gtm,gtm\_slave,pgxc\_ctl}
# chown -R postgres:postgres /var/lib/pgxl

修改host文件

[root@postgres1 ~]# vi /etc/hosts
添加各个节点地址
192.168.2.164 node1
192.168.2.165 node2
192.168.2.166 node3

node1 节点 postgres 用户的环境变量

# su - postgres
$ vi ~/.bash\_profile
export PGUSER=postgres
export PGHOME=/usr/pgxl-10
export PGXC\_CTL\_HOME=/var/lib/pgxl/pgxc\_ctl
export LD\_LIBRARY\_PATH=$PGHOME/lib
export LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:/lib:/usr/lib:/usr/local/lib
export PATH=$PGHOME/bin:$PATH
export TEMP=/tmp
export TMPDIR=/tmp
export PS1="\[\e[32;1m\][\u@\h \W]$>\[\e[0m\]"

node1 上编译安装

$ cd /tmp
$ git clone git://git.postgresql.org/git/postgres-xl.git
$ cd postgres-xl
$ git branch -r
origin/HEAD -> origin/master
origin/XL9\_5\_STABLE
origin/XL\_10\_STABLE
origin/master
origin/xl\_dbt3\_expt
origin/xl\_doc\_update
origin/xl\_test
$ git checkout XL\_10\_STABLE
Branch XL\_10\_STABLE set up to track remote branch XL\_10\_STABLE from origin.
Switched to a new branch 'XL\_10\_STABLE'
$ git status
# On branch XL\_10\_STABLE
nothing to commit, working directory clean
$ ./configure --prefix=/usr/pgxl-10 --with-perl --with-python --with-openssl --with-pam --with-ldap --with-libxml --with-libxslt
$ make
$ make install
$ cd contrib
$ make
$ make install

2.另外在 node2、node3节点上还需要运行如下命令

# su - postgres
$ cd /var/lib/pgxl
$ mkdir {gtm\_proxy}
$ mkdir {coord,coord\_slave,coord\_archlog}
$ mkdir {dn\_master,dn\_slave,dn\_archlog}

node1、node2、node3配置ssh相互免密登录

su - pgxl
ssh-keygen -t rsa
cat ~/.ssh/id\_rsa.pub>> ~/.ssh/authorized\_keys
chmod 600 ~/.ssh/authorized\_keys
将刚生成的认证文件拷贝到另外2台服务器：
scp ~/.ssh/authorized\_keys postgres@192.168.2.165:~/.ssh/
scp ~/.ssh/authorized\_keys postgres@192.168.2.166:~/.ssh/

node1、node2、node3同步下时间

# ntpdate asia.pool.ntp.org

node1,node2,node3 节点修改环境变量（~/.bashrc和~/.bash\_profile都要修改

/etc/hosts
ps -aux | grep pgxl
echo $PGHOME

[postgres@postgres2 ~]$>vi ~/.bashrc
# .bashrc
export PGUSER=postgres
export PGHOME=/usr/pgxl-10
export PGXC\_CTL\_HOME=/var/lib/pgxl/pgxc\_ctl
export LD\_LIBRARY\_PATH=$PGHOME/lib
export LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:/lib:/usr/lib:/usr/local/lib
export PATH=$PGHOME/bin:$PATH
export TEMP=/tmp
export TMPDIR=/tmp
export PS1="\[\e[32;1m\][\u@\h \W]$>\[\e[0m\]"
# Source global definitions
if [ -f /etc/bashrc ]; then
. /etc/bashrc
fi
# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD\_PAGER=
# User specific aliases and functions
##############################################################################
[postgres@postgres2 ~]$>vim ~/.bash\_profile
# .bash\_profile
export PGUSER=postgres
export PGHOME=/usr/pgxl-10
export PGXC\_CTL\_HOME=/var/lib/pgxl/pgxc\_ctl
export LD\_LIBRARY\_PATH=$PGHOME/lib
export LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH}:/lib:/usr/lib:/usr/local/lib
export PATH=$PGHOME/bin:$PATH
export TEMP=/tmp
export TMPDIR=/tmp
export PS1="\[\e[32;1m\][\u@\h \W]$>\[\e[0m\]"
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
. ~/.bashrc
fi
# User specific environment and startup programs
#PATH=$PATH:$HOME/.local/bin:$HOME/bin
#export PATH

不修改环境会出现 command not found 的错误提示

这里是依赖 ~/.bashrc

Current directory: /var/lib/pgxl/pgxc\_ctl
Initialize GTM master
ERROR: target directory (/var/lib/pgxl/gtm) exists and not empty. Skip GTM initilialization
bash: gtm: command not found
bash: gtm\_ctl: command not found
Done.
Start GTM master
bash: gtm\_ctl: command not found
Initialize GTM slave
bash: initgtm: command not found

pgxc\_ctl 生成配置文件

$ which pgxc\_ctl
/usr/pgxl-10/bin/pgxc\_ctl
$ pgxc\_ctl prepare
/bin/bash
Installing pgxc\_ctl\_bash script as /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl\_bash.
ERROR: File "/var/lib/pgxl/pgxc\_ctl/pgxc\_ctl.conf" not found or not a regular file. No such file or directory
Installing pgxc\_ctl\_bash script as /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl\_bash.
Reading configuration using /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl\_bash --home /var/lib/pgxl/pgxc\_ctl --configuration /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl.conf
Finished reading configuration.
\*\*\*\*\*\*\*\* PGXC\_CTL START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
Current directory: /var/lib/pgxl/pgxc\_ctl
$ ls -l /var/lib/pgxl/pgxc\_ctl
total 24
-rw-r--r-- 1 postgres postgres 246 Jul 18 16:42 coordExtraConfig
-rw-r--r-- 1 postgres postgres 17815 Jul 18 16:42 pgxc\_ctl.conf

pgxc\_ctl 修改配置文件: $ vi /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl.conf

pgxc\_ctl 的一些操作

在 node1 节点上操作

#初始化集群（安装）
$ pgxc\_ctl -c /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl.conf init all
#启动集群
$ pgxc\_ctl -c /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl.conf start all
关闭集群
$ pgxc\_ctl -c /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl.conf stop all

重新安装

停止进程ps -aux | grep pgxl
kill -9 xxxx
ERROR: target directory (/var/lib/pgxl/gtm) exists and not empty. Skip GTM initilialization
清理/var/lib/pgxl/gtm路径
node1节点
# cd /var/lib/pgxl
# mkdir {gtm,gtm\_slave,pgxc\_ctl}
node2 node3节点
su - postgres
cd /var/lib/pgxl/gtm\_proxy
$ mkdir gtm\_proxy
$ mkdir {coord,coord\_slave,coord\_archlog}
$ mkdir {dn\_master,dn\_slave,dn\_archlog}
清理路径
问题处理
gtm\_ctl: could not send stop signal (PID: 26076): No such process

EXECUTE DIRECT ON (node2) 'CREATE NODE coord1 WITH (TYPE=''coordinator'', HOST=''node2'', PORT=20004)';

ERROR: PGXC Node node2: object not defined

EXECUTE DIRECT ON (node2) 'CREATE NODE coord2 WITH (TYPE=''coordinator'', HOST=''node3'', PORT=20005)';

ERROR: PGXC Node node2: object not defined

EXECUTE DIRECT ON (node2) 'ALTER NODE node2 WITH (TYPE=''datanode'', HOST=''node2'', PORT=20008, PRIMARY, PREFERRED)';

ERROR: PGXC Node node2: object not defined

EXECUTE DIRECT ON (node2) 'CREATE NODE node3 WITH (TYPE=''datanode'', HOST=''node3'', PORT=20009, PREFERRED)';

ERROR: PGXC Node node2: object not defined

EXECUTE DIRECT ON (node2) 'SELECT pgxc\_pool\_reload()';

ERROR: PGXC Node node2: object not defined

EXECUTE DIRECT ON (node3) 'CREATE NODE coord1 WITH (TYPE=''coordinator'', HOST=''node2'', PORT=20004)';

ERROR: PGXC Node node3: object not defined

EXECUTE DIRECT ON (node3) 'CREATE NODE coord2 WITH (TYPE=''coordinator'', HOST=''node3'', PORT=20005)';

ERROR: PGXC Node node3: object not defined

EXECUTE DIRECT ON (node3) 'CREATE NODE node2 WITH (TYPE=''datanode'', HOST=''node2'', PORT=20008, PRIMARY, PREFERRED)';

ERROR: PGXC Node node3: object not defined

EXECUTE DIRECT ON (node3) 'ALTER NODE node3 WITH (TYPE=''datanode'', HOST=''node3'', PORT=20009, PREFERRED)';

ERROR: PGXC Node node3: object not defined

EXECUTE DIRECT ON (node3) 'SELECT pgxc\_pool\_reload()';

ERROR: PGXC Node node3: object not defined

此问题纠结很久 处理方法

修改host文件:vi /etc/hosts

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.2.164 node1
192.168.2.165 node2
192.168.2.166 node3

修改脚本238 239行为

primaryDatanode=datanode1 # Primary Node.

datanodeNames=(datanode1 datanode2)

问题解决

下面是 的输出日志，记录一下

[postgres@postgres1 ~]$>pgxc\_ctl -c /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl.conf init all
/bin/bash
Installing pgxc\_ctl\_bash script as /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl\_bash.
Installing pgxc\_ctl\_bash script as /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl\_bash.
Reading configuration using /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl\_bash --home /var/lib/pgxl/pgxc\_ctl --configuration /var/lib/pgxl/pgxc\_ctl/pgxc\_ctl.conf
Finished reading configuration.
\*\*\*\*\*\*\*\* PGXC\_CTL START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
Current directory: /var/lib/pgxl/pgxc\_ctl
Initialize GTM master
The files belonging to this GTM system will be owned by user "postgres".
This user must also own the server process.
fixing permissions on existing directory /var/lib/pgxl/gtm ... ok
creating configuration files ... ok
creating control file ... ok
Success.
Done.
Start GTM master
server starting
Initialize GTM slave
The files belonging to this GTM system will be owned by user "postgres".
This user must also own the server process.
fixing permissions on existing directory /var/lib/pgxl/gtm\_slave ... ok
creating configuration files ... ok
creating control file ... ok
Success.
Done.
Start GTM slaveserver starting
Done.
Initialize all the gtm proxies.
Initializing gtm proxy gtm\_proxy1.
Initializing gtm proxy gtm\_proxy2.
gtm\_ctl: could not send stop signal (PID: 20104): No such process
The files belonging to this GTM system will be owned by user "postgres".
This user must also own the server process.
fixing permissions on existing directory /var/lib/pgxl/gtm\_proxy ... ok
creating configuration files ... ok
Success.
gtm\_ctl: could not send stop signal (PID: 12004): No such process
The files belonging to this GTM system will be owned by user "postgres".
This user must also own the server process.
fixing permissions on existing directory /var/lib/pgxl/gtm\_proxy ... ok
creating configuration files ... ok
Success.
Done.
Starting all the gtm proxies.
Starting gtm proxy gtm\_proxy1.
Starting gtm proxy gtm\_proxy2.
server starting
server starting
Done.
Initialize all the coordinator masters.
Initialize coordinator master coord1.
Initialize coordinator master coord2.
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.
The database cluster will be initialized with locale "zh\_CN.UTF-8".
The default database encoding has accordingly been set to "UTF8".
initdb: could not find suitable text search configuration for locale "zh\_CN.UTF-8"
The default text search configuration will be set to "simple".
Data page checksums are disabled.
fixing permissions on existing directory /var/lib/pgxl/coord ... ok
creating subdirectories ... ok
selecting default max\_connections ... 100
selecting default shared\_buffers ... 128MB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... creating cluster information ... ok
syncing data to disk ... ok
freezing database template0 ... ok
freezing database template1 ... ok
freezing database postgres ... ok
WARNING: enabling "trust" authentication for local connections
You can change this by editing pg\_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
Success.
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.
The database cluster will be initialized with locale "zh\_CN.UTF-8".
The default database encoding has accordingly been set to "UTF8".
initdb: could not find suitable text search configuration for locale "zh\_CN.UTF-8"
The default text search configuration will be set to "simple".
Data page checksums are disabled.
fixing permissions on existing directory /var/lib/pgxl/coord ... ok
creating subdirectories ... ok
selecting default max\_connections ... 100
selecting default shared\_buffers ... 128MB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... creating cluster information ... ok
syncing data to disk ... ok
freezing database template0 ... ok
freezing database template1 ... ok
freezing database postgres ... ok
WARNING: enabling "trust" authentication for local connections
You can change this by editing pg\_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
Success.
Done.
Starting coordinator master.
Starting coordinator master coord1
Starting coordinator master coord2
2019-06-11 14:08:22.834 CST [21489] LOG: listening on IPv4 address "0.0.0.0", port 20004
2019-06-11 14:08:22.834 CST [21489] LOG: listening on IPv6 address "::", port 20004
2019-06-11 14:08:22.837 CST [21489] LOG: listening on Unix socket "/tmp/.s.PGSQL.20004"
2019-06-11 14:08:22.846 CST [21489] LOG: redirecting log output to logging collector process
2019-06-11 14:08:22.846 CST [21489] HINT: Future log output will appear in directory "pg\_log".
2019-06-11 14:08:22.801 CST [13773] LOG: listening on IPv4 address "0.0.0.0", port 20005
2019-06-11 14:08:22.801 CST [13773] LOG: listening on IPv6 address "::", port 20005
2019-06-11 14:08:22.804 CST [13773] LOG: listening on Unix socket "/tmp/.s.PGSQL.20005"
2019-06-11 14:08:22.814 CST [13773] LOG: redirecting log output to logging collector process
2019-06-11 14:08:22.814 CST [13773] HINT: Future log output will appear in directory "pg\_log".
Done.
Initialize all the coordinator slaves.
Initialize the coordinator slave coord1.
Initialize the coordinator slave coord2.
Done.
Starting all the coordinator slaves.
Starting coordinator slave coord1.
Starting coordinator slave coord2.
2019-06-11 14:08:28.443 CST [14091] LOG: listening on IPv4 address "0.0.0.0", port 20004
2019-06-11 14:08:28.443 CST [14091] LOG: listening on IPv6 address "::", port 20004
2019-06-11 14:08:28.446 CST [14091] LOG: listening on Unix socket "/tmp/.s.PGSQL.20004"
2019-06-11 14:08:28.457 CST [14091] LOG: redirecting log output to logging collector process
2019-06-11 14:08:28.457 CST [14091] HINT: Future log output will appear in directory "pg\_log".
2019-06-11 14:08:28.464 CST [21806] LOG: listening on IPv4 address "0.0.0.0", port 20005
2019-06-11 14:08:28.464 CST [21806] LOG: listening on IPv6 address "::", port 20005
2019-06-11 14:08:28.472 CST [21806] LOG: listening on Unix socket "/tmp/.s.PGSQL.20005"
2019-06-11 14:08:28.484 CST [21806] LOG: redirecting log output to logging collector process
2019-06-11 14:08:28.484 CST [21806] HINT: Future log output will appear in directory "pg\_log".
Done
Initialize all the datanode masters.
Initialize the datanode master datanode1.
Initialize the datanode master datanode2.
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.
The database cluster will be initialized with locale "zh\_CN.UTF-8".
The default database encoding has accordingly been set to "UTF8".
initdb: could not find suitable text search configuration for locale "zh\_CN.UTF-8"
The default text search configuration will be set to "simple".
Data page checksums are disabled.
fixing permissions on existing directory /var/lib/pgxl/dn\_master ... ok
creating subdirectories ... ok
selecting default max\_connections ... 100
selecting default shared\_buffers ... 128MB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... creating cluster information ... ok
syncing data to disk ... ok
freezing database template0 ... ok
freezing database template1 ... ok
freezing database postgres ... ok
WARNING: enabling "trust" authentication for local connections
You can change this by editing pg\_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
Success.
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.
The database cluster will be initialized with locale "zh\_CN.UTF-8".
The default database encoding has accordingly been set to "UTF8".
initdb: could not find suitable text search configuration for locale "zh\_CN.UTF-8"
The default text search configuration will be set to "simple".
Data page checksums are disabled.
fixing permissions on existing directory /var/lib/pgxl/dn\_master ... ok
creating subdirectories ... ok
selecting default max\_connections ... 100
selecting default shared\_buffers ... 128MB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... creating cluster information ... ok
syncing data to disk ... ok
freezing database template0 ... ok
freezing database template1 ... ok
freezing database postgres ... ok
WARNING: enabling "trust" authentication for local connections
You can change this by editing pg\_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
Success.
Done.
Starting all the datanode masters.
Starting datanode master datanode1.
Starting datanode master datanode2.
2019-06-11 14:08:35.227 CST [22292] LOG: listening on IPv4 address "0.0.0.0", port 20008
2019-06-11 14:08:35.227 CST [22292] LOG: listening on IPv6 address "::", port 20008
2019-06-11 14:08:35.327 CST [22292] LOG: listening on Unix socket "/tmp/.s.PGSQL.20008"
2019-06-11 14:08:35.405 CST [22292] LOG: redirecting log output to logging collector process
2019-06-11 14:08:35.405 CST [22292] HINT: Future log output will appear in directory "pg\_log".
2019-06-11 14:08:35.196 CST [14575] LOG: listening on IPv4 address "0.0.0.0", port 20009
2019-06-11 14:08:35.197 CST [14575] LOG: listening on IPv6 address "::", port 20009
2019-06-11 14:08:35.296 CST [14575] LOG: listening on Unix socket "/tmp/.s.PGSQL.20009"
2019-06-11 14:08:35.376 CST [14575] LOG: redirecting log output to logging collector process
2019-06-11 14:08:35.376 CST [14575] HINT: Future log output will appear in directory "pg\_log".
Done.
Initialize all the datanode slaves.
Initialize datanode slave datanode1
Initialize datanode slave datanode2
Starting all the datanode slaves.
Starting datanode slave datanode1.
Starting datanode slave datanode2.
2019-06-11 14:08:42.349 CST [14892] LOG: listening on IPv4 address "0.0.0.0", port 20008
2019-06-11 14:08:42.349 CST [14892] LOG: listening on IPv6 address "::", port 20008
2019-06-11 14:08:42.352 CST [14892] LOG: listening on Unix socket "/tmp/.s.PGSQL.20008"
2019-06-11 14:08:42.361 CST [14892] LOG: redirecting log output to logging collector process
2019-06-11 14:08:42.361 CST [14892] HINT: Future log output will appear in directory "pg\_log".
2019-06-11 14:08:42.377 CST [22609] LOG: listening on IPv4 address "0.0.0.0", port 20009
2019-06-11 14:08:42.377 CST [22609] LOG: listening on IPv6 address "::", port 20009
2019-06-11 14:08:42.379 CST [22609] LOG: listening on Unix socket "/tmp/.s.PGSQL.20009"
2019-06-11 14:08:42.389 CST [22609] LOG: redirecting log output to logging collector process
2019-06-11 14:08:42.389 CST [22609] HINT: Future log output will appear in directory "pg\_log".
Done.
ALTER NODE coord1 WITH (HOST='node2', PORT=20004);
ALTER NODE
CREATE NODE coord2 WITH (TYPE='coordinator', HOST='node3', PORT=20005);
CREATE NODE
CREATE NODE datanode1 WITH (TYPE='datanode', HOST='node2', PORT=20008, PRIMARY, PREFERRED);
CREATE NODE
CREATE NODE datanode2 WITH (TYPE='datanode', HOST='node3', PORT=20009);
CREATE NODE
SELECT pgxc\_pool\_reload();
pgxc\_pool\_reload
------------------
t
(1 row)
CREATE NODE coord1 WITH (TYPE='coordinator', HOST='node2', PORT=20004);
CREATE NODE
ALTER NODE coord2 WITH (HOST='node3', PORT=20005);
ALTER NODE
CREATE NODE datanode1 WITH (TYPE='datanode', HOST='node2', PORT=20008, PRIMARY);
CREATE NODE
CREATE NODE datanode2 WITH (TYPE='datanode', HOST='node3', PORT=20009, PREFERRED);
CREATE NODE
SELECT pgxc\_pool\_reload();
pgxc\_pool\_reload
------------------
t
(1 row)
Done.
EXECUTE DIRECT ON (datanode1) 'CREATE NODE coord1 WITH (TYPE=''coordinator'', HOST=''node2'', PORT=20004)';
EXECUTE DIRECT
EXECUTE DIRECT ON (datanode1) 'CREATE NODE coord2 WITH (TYPE=''coordinator'', HOST=''node3'', PORT=20005)';
EXECUTE DIRECT
EXECUTE DIRECT ON (datanode1) 'ALTER NODE datanode1 WITH (TYPE=''datanode'', HOST=''node2'', PORT=20008, PRIMARY, PREFERRED)';
EXECUTE DIRECT
EXECUTE DIRECT ON (datanode1) 'CREATE NODE datanode2 WITH (TYPE=''datanode'', HOST=''node3'', PORT=20009, PREFERRED)';
EXECUTE DIRECT
EXECUTE DIRECT ON (datanode1) 'SELECT pgxc\_pool\_reload()';
pgxc\_pool\_reload
------------------
t
(1 row)
EXECUTE DIRECT ON (datanode2) 'CREATE NODE coord1 WITH (TYPE=''coordinator'', HOST=''node2'', PORT=20004)';
EXECUTE DIRECT
EXECUTE DIRECT ON (datanode2) 'CREATE NODE coord2 WITH (TYPE=''coordinator'', HOST=''node3'', PORT=20005)';
EXECUTE DIRECT
EXECUTE DIRECT ON (datanode2) 'CREATE NODE datanode1 WITH (TYPE=''datanode'', HOST=''node2'', PORT=20008, PRIMARY, PREFERRED)';
EXECUTE DIRECT
EXECUTE DIRECT ON (datanode2) 'ALTER NODE datanode2 WITH (TYPE=''datanode'', HOST=''node3'', PORT=20009, PREFERRED)';
EXECUTE DIRECT
EXECUTE DIRECT ON (datanode2) 'SELECT pgxc\_pool\_reload()';
pgxc\_pool\_reload
------------------
t
(1 row)
Done.

**四.验证**

登录 node2节点的 coordinator，发现都不用再手动 create node

$ psql -p 20004
psql (PGXL 10alpha2, based on PG 10.3 (Postgres-XL 10alpha2))
Type "help" for help.
postgres=#
postgres=# select \* from pgxc\_node;
node\_name | node\_type | node\_port | node\_host | nodeis\_primary | nodeis\_preferred | node\_id
-----------+-----------+-----------+-----------+----------------+------------------+-------------
coord1 | C | 20004 | node2 | f | f | 1885696643
coord2 | C | 20005 | node3 | f | f | -1197102633
datanode1 | D | 20008 | node2 | t | t | -927910690
datanode2 | D | 20009 | node3 | f | f | 914546798
(4 rows)

再登录下 node3节点的 coordinator。

$ psql -p 20005
psql (PGXL 10alpha2, based on PG 10.3 (Postgres-XL 10alpha2))
Type "help" for help.
postgres=# select \* from pgxc\_node;
node\_name | node\_type | node\_port | node\_host | nodeis\_primary | nodeis\_preferred | node\_id
-----------+-----------+-----------+-----------+----------------+------------------+-------------
coord1 | C | 20004 | node2 | f | f | 1885696643
coord2 | C | 20005 | node3 | f | f | -1197102633
datanode1 | D | 20008 | node2 | t | f | -927910690
datanode2 | D | 20009 | node3 | f | t | 914546798
(4 rows)

在任意一个coordinator执行如下操作

postgres=# create database peiybdb;
postgres=# \c peiybdb
peiybdb=# create table tmp\_t0(c0 varchar(100),c1 varchar(100));
peiybdb=# insert into tmp\_t0(c0,c1) SELECT id::varchar,md5(id::varchar) FROM generate\_series(1,10000) as id;
INSERT 0 10000
peiybdb=# select xc\_node\_id,count(1) from tmp\_t0 group by xc\_node\_id;
xc\_node\_id | count
------------+-------
-927910690 | 5081
914546798 | 4919
(2 rows)

**五.参考文章**

<https://blog.csdn.net/ctypyb2002/article/details/81104535#comments>

<https://www.jianshu.com/p/41d857a5d743>

<https://blog.csdn.net/linuxchen/article/details/81509397>

<https://www.cnblogs.com/lottu/p/5646486.html>

官方文档

<https://www.postgres-xl.org/documentation/>

<https://www.postgres-xl.org/documentation/runtime.html>

<https://www.postgres-xl.org/documentation/runtime-config.html>

<https://www.postgres-xl.org/documentation/pgxc-ctl.html>

<https://www.postgres-xl.org/>

<https://www.postgres-xl.org/overview/>

<https://www.postgres-xl.org/download/>

https://git.postgresql.org/gitweb/?p=postgres-xl.git;a=summary