GNU/Linux Administration - Support #697

Install Pgpool2 on Debian

11/18/2015 12:22 PM - Daniel Curtis

Status:	Closed	**Start date:**	11/18/2015
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	60%
Category:	Database Server	**Estimated time:**	2.00 hours
Target version:	Debian	**Spent time:**	4.50 hours

Description

This is a guide for setting up pgpool2 to handle a pool of postgresql databases on Debian 8.

Prepare the Environment

- Make sure the system is up to date:

```
apt-get update && apt-get upgrade
```

Install Pgpool2

- Install pgpool2:

```
apt-get install postgresql postgresql-9.4-pgpool2 rsync ssh sudo pgpool2
```

- Edit pg_hba.conf to enable database access via the local socket:

```
nano /etc/postgresql/9.4/main/pg_hba.conf
```

  - Replace peer with trust

```
local   all         all                                     trust
```

- Edit the main postgresql config:

```
nano /etc/postgresql/9.4/main/postgresql.conf
```

  - And change the listen_addresses parameter:

```
listen_addresses = '*'

max_wal_senders = 1
wal_level = hot_standby
archive_mode = on
archive_command = 'test ! -f /var/lib/postgresql/9.4/main/archive_log/backup_in_progress |
| (test -f /var/lib/postgresql/9.4/main/archive_log/%f || cp %p /var/lib/postgresql/9.4/ma
in/archive_log/%f)'
```

- Install the archive_log command:

```
install -o postgres -g postgres -m 700 -d /var/lib/postgresql/9.4/main/archive_log
```

- Restart postgresql:

```
systemctl restart postgresql
```

- Log into the postgres database as the admin:

```
su - postgres -c 'psql template1'
```

  - And create the pgpool_recovery extension:

    ```
    CREATE EXTENSION "pgpool_recovery"
    \q
    exit
    ```

- Setup local passwordless login for the postgres user

```
su - postgres
ssh-keygen -f /var/lib/postgresql/.ssh/id_rsa -N ''
cat .ssh/id_rsa.pub >> .ssh/authorized_keys
exit
```

- Setup access remote databases with pg_basebackup non-interactively by creating a password file:

```
su - postgres
echo '*:*:*:postgres:SuperSecretPassword' > .pgpass
chmod 0600 .pgpass
exit
```

- Allow the postgres user to restart postgresql without a password:

```
echo 'postgres ALL=(ALL:ALL) NOPASSWD:/bin/systemctl start postgresql.service' >> /etc/sudoers
.d/pgpool-postgres
echo 'postgres ALL=(ALL:ALL) NOPASSWD:/bin/systemctl stop postgresql.service' >> /etc/sudoers.
d/pgpool-postgres
```

- Edit pg_hba.conf to enable database access via the local socket:

```
nano /etc/postgresql/9.4/main/pg_hba.conf
```

  - And add the postgres user replication access using md5 authentication:

    ```
    local   all             all                                     trust
    host    replication     postgres     .example.com               md5
    ```

- After enabling password-based authentication, we need to set a password for the postgres user which we'll use for making the base backup:

```
su postgres -c psql
echo "ALTER USER postgres WITH PASSWORD 'SuperSecretPassword';"
```

Configure Pgpool2

- Create a pgpool2 config from the installed examples:

```
gunzip -c /usr/share/doc/pgpool2/examples/pgpool.conf.sample-replication.gz > /etc/pgpool2/pgpool.conf
```

- Set a password for the pcp admin user:

```
echo "postgres:$(pg_md5 SuperSecretPassword)" >> /etc/pgpool2/pcp.conf
```

- In replication mode, when the client should authenticate towards the PostgreSQL database, configure pgpool2 to use password-based authentication:

```
sed -i 's/trust$/md5/g' /etc/pgpool2/pool_hba.conf
sed -i 's/\(enable_pool_hba =\) off/\1 on/g' /etc/pgpool2/pgpool.conf
```

- Create all the usernames and passwords that will be used to connect to pgpool2:

```
touch pool_passwd
chown postgres.postgres pool_passwd
pg_md5 -m -u pg2user SuperSecretPg2Password
```

- My use case has servers on separate networks, so it is advisable to turn off load_balance_mode, otherwise queries will be sent to all healthy backends. In addition, a higher weight will be assigned to the backend which is on the same machine as pgpool2, so read-only queries are sent to the local backend only.

```
sed -i 's/^load_balance_mode = on/load_balance_mode = off/g' /etc/pgpool2/pgpool.conf
```

- Comment out any currently defined backends:

```
sed -i 's/^\(backend_\)/# \1/g' /etc/pgpool2/pgpool.conf
```

- Edit the pgpool2 config file:

```
nano /etc/pgpool2/pgpool.conf
```

  - And add the following to the end of the config file to define the **pg1.example.com** backend host:

    ```
    backend_hostname0 = 'pg1.example.com'
    backend_port0 = 5432
    backend_weight0 = 2
    backend_data_directory0 = '/var/lib/postgresql/9.4/main'
    ```

  - And add the following to the end of the config file to define the **pg2.example.com** backend host:

    ```
    backend_hostname1 = 'pg2.example.com'
    backend_port1 = 5432
    backend_weight1 = 1
    backend_data_directory1 = '/var/lib/postgresql/9.4/main'
    ```

Configuring recovery

- Define a couple of shell scripts that handle the details of how the recovery is performed:

```
sed -i 's/^\(recovery_\|client_idle_limit_in_recovery\)/# \1/g' /etc/pgpool2/pgpool.conf
```

- Edit the pgpool2 config:

```
nano /etc/pgpool2/pgpool.conf
```

  - And add the following:

```
recovery_user = 'postgres'
recovery_password = 'SuperSecretPassword'

recovery_1st_stage_command = '1st_stage.sh'
recovery_2nd_stage_command = '2nd_stage.sh'

client_idle_limit_in_recovery = -1
```

- The 1st_stage.sh script logs into the backend that should be recovered and uses pg_basebackup to copy a full backup from the master(primary) backend. It also sets up the recovery.conf which will be used by PostgreSQL when starting up:

```
nano /var/lib/postgresql/9.4/main/1st_stage.sh
```

  - And add the following:

```
#!/bin/sh
TS=$(date +%Y-%m-%d_%H-%M-%S)
MASTER_HOST=$(hostname -f)
MASTER_DATA=$1
RECOVERY_TARGET=$2
RECOVERY_DATA=$3

# Move the PostgreSQL data directory out of our way.
ssh -T $RECOVERY_TARGET \
    "[ -d $RECOVERY_DATA ] && mv $RECOVERY_DATA $RECOVERY_DATA.$TS"

# We only use archived WAL logs during recoveries, so delete all
# logs from the last recovery to limit the growth.
rm $MASTER_DATA/archive_log/*

# With this file present, our archive_command will actually
# archive WAL files.
touch $MASTER_DATA/archive_log/backup_in_progress

# Perform a backup of the database.
ssh -T $RECOVERY_TARGET \
    "pg_basebackup -h $MASTER_HOST -D $RECOVERY_DATA --xlog"

# Configure the restore_command to use the archive_log WALs we'll copy
# over in 2nd_stage.sh.
echo "restore_command = 'cp $RECOVERY_DATA/archive_log/%f %p'" | \
    ssh -T $RECOVERY_TARGET "cat > $RECOVERY_DATA/recovery.conf"
```

- Create the 2nd_stage.sh script:

```
nano /var/lib/postgresql/9.4/main/2nd_stage.sh
```

- And add the following:

```
#! /bin/sh
MASTER_DATA=$1
RECOVERY_TARGET=$2
RECOVERY_DATA=$3
port=5432

# Force to flush current value of sequences to xlog
psql -p $port -t -c 'SELECT datname FROM pg_database WHERE NOT datistemplate AND datallowc
onn' template1|
while read i
do
  if [ "$i" != "" ];then
    psql -p $port -c "SELECT setval(oid, nextval(oid)) FROM pg_class WHERE relkind = 'S'"
$i
  fi
done

# Flush all transactions to disk. Since pgpool stopped all connections,
# there cannot be any data that does not reside on disk until the
# to-be-recovered host is back on line.
psql -p $port -c "SELECT pgpool_switch_xlog('$MASTER_DATA/archive_log')" template1

# Copy over all archive logs at once.
rsync -avx --delete $MASTER_DATA/archive_log/ \
    $RECOVERY_TARGET:$RECOVERY_DATA/archive_log/

# Delete the flag file to disable WAL archiving again.
rm $MASTER_DATA/archive_log/backup_in_progress
```

- Also create a pgpool_remote_start script:

```
nano /var/lib/postgresql/9.4/main/pgpool_remote_start
```

- And add the following:

```
#!/bin/sh
ssh $1 sudo systemctl start postgresql.service
```

- Make the scripts executable:

```
chmod +x /var/lib/postgresql/9.4/main/1st_stage.sh
chmod +x /var/lib/postgresql/9.4/main/2nd_stage.sh
chmod +x /var/lib/postgresql/9.4/main/pgpool_remote_start
```

- Now start pgpool2 and verify that it works and can access the first node. The pcp_node_count command should return an integer number like “2”. The psql command should be able to connect and you should see your database tables when using \d.

```
systemctl restart pgpool2.service
pcp_node_count 10 localhost 9898 postgres SuperSecretPassword
psql -p 5433 -U pg2user pg2db
```

Install pgpoolAdmin

- Install nginx and php-fpm

```
apt-get install nginx php5-fpm php5-pgsql curl
```

- Download and extract pgpoolAdmin:

```
cd /var/www
curl -L http://www.pgpool.net/download.php?f=pgpoolAdmin-3.4.1.tar.gz -o pgpoolAdmin-3.4.1.tar
.gz
tar xzf pgpoolAdmin-3.4.1.tar.gz
mv pgpoolAdmin-3.4.1 pgpooladmin
```

- The installation will now complete. Now create a file to enable phppgadmin:

```
nano /etc/nginx/sites-enabled/phppgadmin
```

  - And add the following:

```
server {
    listen          8080;
    server_name     localhost;

    location / {
        root    /var/www/pgpooladmin;
        index   index.php;
    }

    location ~ \.php$ {
        root            /var/www/pgpooladmin;
        fastcgi_pass unix:/var/run/php5-fpm.sock;
        fastcgi_index   index.php;
        fastcgi_param   SCRIPT_FILENAME  $document_root$fastcgi_script_name;
        include         fastcgi_params;
    }
}
```

- Set the ownership of the pgpooladmin user to the web user:

```
chown -R www-data:www-data /var/www/pgpooladmin/
```

- Change the ownership of the pgpool and pcp configs:

```
chown www-data /etc/pgpool2/pgpool.conf
chmod 644 /etc/pgpool2/pgpool.conf
chown www-data /etc/pgpool2/pcp.conf
chmod 644 /etc/pgpool2/pcp.conf
```

- Restart nginx:

```
systemctl restart nginx
```

- Open http://pgpooladmin.example.com/install/index.php in a web browser to finish the installation
  - Change the pgpool.conf file path to /etc/pgpool2/pgpool.conf
  - Change the pcp.conf file path to /etc/pgpool2/pcp.conf
  - Change the pgpool command path to /usr/sbin/pgpool
  - Change the PCP directory path to /usr/sbin/pgpool

- When the installation is finished, remove the install directory:

```
rm -rf /var/www/pgpooladmin/install
```

Resources

- http://michael.stapelberg.de/Artikel/replicated_postgresql_with_pgpool/
- http://www.pgpool.net/docs/latest/pgpool-en.html
- http://cloud-engineering.forthscale.com/2013/03/how-to-install-pgpool-ii-on-postgresql.html
- http://www.keyup.eu/en/blog/89-replication-and-load-balancing-with-postgresql-and-pgpool2

History

#1 - 11/18/2015 01:31 PM - Daniel Curtis

- Description updated

- Status changed from New to In Progress

- % Done changed from 0 to 30

#2 - 11/18/2015 03:52 PM - Daniel Curtis

- Description updated

- % Done changed from 30 to 60

#3 - 06/04/2017 09:11 PM - Daniel Curtis

- Status changed from In Progress to Closed