COMO FAZER REPLICACAO MASTER-SLAVE DE BANCO DE DADOS POSTGRESQL COM GATILHO FAILOVER

Hudson Murilo dos Santos

Minha opinião sobre o porque fazer replicação via archives e não usando slony e outras ferramentas próprias para replicação de PostgreSQL:

Sou da opinião de que se quer fazer funcionar uma replicação de banco de dados, deve-se conhecer na raiz como tudo funciona, com os recursos do próprio banco de dados. Além de a documentação ser melhor e poder ter um melhor controle sobre definições como RPO e RTO.
Depois que se aprende a fazer tudo "na mão grande" é beeeem mais tranquilo de aprender uma ferramenta e aplicar nela os conceitos que você já conheceu na base.

Definições iniciais para entendimento global da solução:

1) Usaremos no ambiente de teste um Fedora 13 x86_64 e um CentOS 5.4 x86_64

2) Por padrão as instâncias do PostgreSQL ficam dentro do "diretório do cluster" que é o /var/lib/pgsql/data/

3) Porta que o PostgreSQL escuta é a 5432

4) Arquivo geral de configuração do PostgreSQL é o /var/lib/pgsql/data/postgresql.conf

5) Arquivo de permissoes de conexao de usuarios de rede e autenticação: /var/lib/pgsql/data/pg_hba.conf

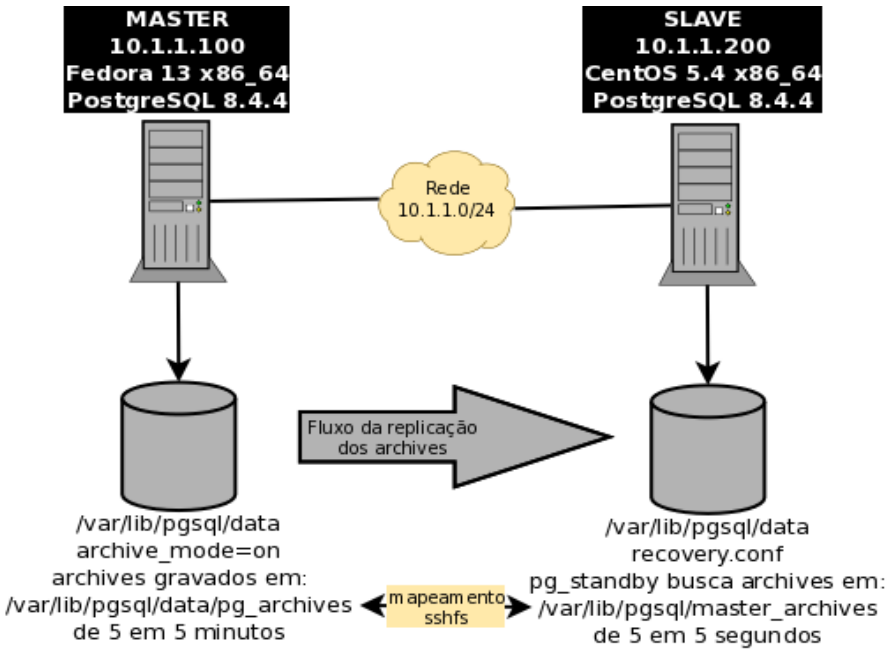
6) Log de startup do banco fica em: /var/lib/pgsql/pgstartup.log

7) Log geral do postgresql fica nomeado por semana dentro de: /var/lib/pgsql/data/pg_log/

8) O script de gerenciamento do serviço é o /etc/init.d/postgresql

9) O diretório padrão para backups do PostgreSQL é o /var/lib/pgsql/backups

10) O diretório /var/lib/pgsql/data/pg_archives existe na máquina MASTER e será compartilhado via SSHFS para a máquina SLAVE



Após o endendimento e visão geral da soluçao como um todo (overview), podemos prosseguir.

A primeira coisa é dispor de 02 computadores para testes e instalar 02 servidores identicos com sistema operacional Linux, e com a mesma versão do PostgreSQL instalada (neste exemplo uso a versão 8.4.4).

Vamos ao passo a passo sempre pensando que a partir de agora os 02 servidores identicos não serão mais identicos. Teremos duas coisas diferentes: MASTER e SLAVE, que é o mesmo que dizer PRODUÇÃO e ALTA-DISPONIBILIDADE, ou ainda TITULAR e RESERVA... é tudo a mesma coisa.

As configurações do MASTER terão características de banco MASTER e as configurações da SLAVE terão características de banco SLAVE.

Vamos ao step-by-step:

#=====================================
 # CONFIGURACOES DO POSTGRESQL MASTER:
#=====================================

Cabe aqui citar um conceito para entendimento:

Um servidor de banco de dados comum, sem replicação, tem o seguinte comportamento: Recebe as transações dos clientes e grava em seu banco de dados para posterior consulta, certo?

Um servidor de banco de dados MASTER, que faz parte de uma solução de replicação, tem um papel um pouco diferente: Ele recebe as transações dos clientes, grava em seu banco normalmente, e também gera um log de transação chamado "archive log" contendo essas mesmas transações que acabou de gravar em seu banco. No arquivo de configuração do MASTER, podemos especificar em qual diretório o banco irá disponibilizar estes archives, que posteriormente serão usados pela máquina SLAVE pois são neles que encontram-se as informações que precisam ser replicadas.

Lembrando que tudo issso é conceito geral existente em qualquer banco de dados de replicação, não somente no PostgreSQL..

Feito essas configurações, o banco já pode ser iniciado.. sabendo que quando iniciamos o serviço do banco de dados pela primeira vez ou quando removemos a pasta "data", seu diretório "data" pode ser criado utilizando o comando initdb conforme abaixo:

```
[root@vostrolab ~]# /etc/init.d/postgresql start
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory

/var/lib/pgsql/data is missing. Use "service postgresql initdb" to initialize the cluster first.
                                        [FAILED]
[root@vostrolab ~]# /etc/init.d/postgresql initdb
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
Initializing database:                                 [  OK  ]
[root@vostrolab ~]#
```

Por padrão, um banco PostgreSQL não vem programado para gerar archivelogs. Para que ela tenha esse comportamento de gerar archivelogs, é necessário acertar alguns parâmetros no arquivo postgresql.conf:

```
archive_mode = on
archive_command = 'cp -i %p /var/lib/pgsql/data/pg_archives/%f </dev/null && echo "[`date`] Archive %p gerado com sucesso..." >> /var/lib/pgsql/data/pg_log/archive_command.log'
archive_timeout = 300
```

Configurar também o arquivo postgresql.conf para escutar em todos os endereços de rede:

```
listen_addresses = '*'
```

Configurar o arquivo pg_hba.conf para aceitar conexoes da rede local autenticando md5:

```
# IPv4 local connections:
host    all         all         10.1.1.0/24          md5
```

Reiniciar o banco:

```
[root@vostrolab ~]# /etc/init.d/postgresql restart
    Stopping postgresql service:                           [  OK  ]
    Starting postgresql service:                           [  OK  ]
```

Configurar a role postgres com a senha postgres apenas por segurança:

```
postgres=# alter role postgres with password 'postgres';
ALTER ROLE
postgres=# commit
```

Neste momento o banco MASTER está pronto para ser replicado. Mas cabe algumas definições para entendimento.

Os archives em processo transacional ficam em: **/var/lib/pgsql/data/pg_xlog/**
Os archives prontos para serem replicados ficam em: **/var/lib/pgsql/data/pg_archives/**
Um arquivo com o mesmo nome (veja bem, apenas o nome) dos archives que o banco ja terminou de transacionar fica em: **/var/lib/pgsql/data/pg_xlog/archive_status**
O ultimo archive transacionado e disponivel eh mostrado no processo do PostgreSQL archiver conforme abaixo:

```
-bash-3.2$ ps aux | grep -v grep | grep \"archiver process\"
postgres  7125  0.0  0.0 159888  1096 ?        Ss   Jul18   0:03 postgres: archiver process   last was 000000010000000010000009E
```

O comando para forçar a geração de um archivelog é executado dentro do banco conforme abaixo:

```
postgres=# select pg_switch_xlog();
 pg_switch_xlog
----------------
 1/F1000188
(1 row)
```

Outras formas de se executar o mesmo comando:

```
echo "SELECT pg_switch_xlog();" | psql -U postgres
psql -c "SELECT pg_switch_xlog();"
```

Vai da preferência de cada administrador.

Agora nosso banco MASTER precisa de um clone seu na maquina SLAVE para então iniciar a

replicação apenas da diferença entre um e outro. Esse é um processo delicado e requer certa atenção.

Para gerar um backup do banco master e replicá-lo para a máquina slave, não podemos fazer usando o método DUMP, temos que fazer backup de DATAFILES conforme instruções abaixo:

1º Com o banco no ar, conectar nele e marcá-lo como "em processo de backup":

```
[root@vostrolab ~]# su - postgres -c "psql -c \"select pg_start_backup('hot_backup');\""
```

2º Gerar pacote compactado .tar.bz2 com o conteúdo do banco de dados:

```
[root@vostrolab ~]# tar -cjf /var/lib/pgsql/backups/data_master.tar.bz2 /var/lib/pgsql/data/
```

3º Conectar no banco e marcá-lo como "fim do backup":

```
[root@vostrolab ~]# su - postgres -c "psql -c \"select pg_stop_backup();\""
```

4º Acertar permissões do diretório onde os backups são gerados:

```
chown -R postgres:postgres /var/lib/pgsql/backups
```

5º Beleza, podemos copiar o backup para a máquina slave:

```
[root@vostrolab ~]# scp -c arcfour /var/lib/pgsql/backups/data_master.tar.bz2
postgres@$ip_do_slave:/var/lib/pgsql/backups
```

Por aqui terminaram nossas atividades na máquina MASTER. Partimos então pra SLAVE...

#====================================
 # CONFIGURACOES DO POSTGRESQL SLAVE:
#====================================

A máquina SLAVE, além de ter o banco PostgreSQL 8.4 instalado, precisamos também de alguns pacotes de contribuintes para que a replicação funcione (o PostgreSQL gosta de focar e por algum motivo nao junta esses pacotes de replicacao :/).

O principal binário pra gente neste momento é o **pg_standby**.

Geralmente ele vem no pacote RPM postgresql-contrib.x86_64 (yum -y install postgresql-contrib.x86_64) e no sistema operacional, fica armazenado em /usr/bin/pg_standby.

As demais soluções do postgresql-contrib ficam armazenadas em /usr/share/pgsql/contrib

Também vamos usar o SSHFS para mapear um diretório do servidor MASTER aqui localmente na STANDBY.
Segue lista resumida de pacotes que precisamos em nossa máquina STANDBY:

```
postgresql84-contrib-8.4.4-1.el5_5.1
postgresql84-libs-8.4.4-1.el5_5.1
postgresql84-8.4.4-1.el5_5.1
fuse-sshfs-2.2-1.el5.rf
```

Feito essas instalações, o banco já pode ser iniciado.. sabendo que quando iniciamos o serviço do banco de dados pela primeira vez ou quando removemos a pasta "data", seu diretório "data" pode ser criado utilizando o comando initdb conforme abaixo:

```
[root@oracle02 ~]# /etc/init.d/postgresql start
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory

/var/lib/pgsql/data is missing. Use "service postgresql initdb" to initialize the cluster first.
                                           [FAILED]
[root@oracle02 ~]# /etc/init.d/postgresql initdb
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
Initializing database:                                 [  OK  ]
[root@oracle02 ~]#
```

Com o banco instalado, podemos configurar o arquivo postgresql.conf para escutar em todos os endereços de rede:

```
listen_addresses = '*'
```

Configurar o arquivo pg_hba.conf para aceitar conexoes da rede local autenticando md5:

```
# IPv4 local connections:
host    all         all         10.1.1.0/24          md5
```

Reiniciar o banco:

```
[root@vostrolab ~]# /etc/init.d/postgresql restart
    Stopping postgresql service:                           [  OK  ]
    Starting postgresql service:                          [  OK  ]
```

Configurar a role postgres com a senha postgres apenas por segurança:

```
postgres=# alter role postgres with password 'postgres';
ALTER ROLE
postgres=# commit
```

Gerar aqui na SLAVE a chave ssh com usuario postgres, para poder estabelecer confiança SSH entre as duas maquinas e poder montar com sshfs sem senha(acertar permissoes 700 e 600 conforme abaixo):

```
-bash-3.2$ whoami
postgres
-bash-3.2$ ssh-keygen -t rsa
-bash-3.2$ cat /var/lib/pgsql/.ssh/id_rsa.pub | ssh root@ip_maquina_master "mkdir -p ~postgres/.ssh ; chmod 700 ~postgres/.ssh; cat - > ~postgres/.ssh/authorized_keys; chmod 600 ~postgres/.ssh/authorized_keys; chown postgres:postgres ~postgres/.ssh/authorized_keys"
```

Abaixo estão as permissoes que devem ter ficado para os diretorios/arquivos que você criou na maquina MASTER com o comando acima (é bom conferir):

```
drwx------   2 postgres postgres 4096 Jul 19 14:20 .ssh
-rw-------   1 postgres postgres  399 Jul 19 14:20 authorized_keys
```

Permitir usuario postgres no sudoers sem senha para os binarios do sshfs e fusermount (utilize o comando visudo para edicao do sudoers):

```
postgres ALL=(ALL)          NOPASSWD: /bin/fusermount , /usr/bin/sshfs
```

Criar diretorio /var/lib/pgsql/master_archives/ para servir de ponto de montagem, e então mapear via sshfs:

```
sudo sshfs -o allow_other postgres@10.1.1.100:/var/lib/pgsql/data/pg_archives /var/lib/pgsql/master_archives/
```

Parar o banco:

```
-bash-3.2$ /etc/init.d/postgresql stop
```

Substituir o banco atual pelo backup da base MASTER (aquele que copiamos pra maquina SLAVE no finalzinho das configurações da maquina MASTER) sempre fazendo backup pelomenos dos arquivos de configuração que já configuramos:

```
-bash-3.2$ cp -rfp /var/lib/pgsql/data /var/lib/pgsql/backups/data_anterior_slave
-bash-3.2$ rm -rf /var/lib/pgsql/data
-bash-3.2$ tar -xjf /var/lib/pgsql/backups/data_master.tar.bz2 --strip-components 3 -C /var/lib/pgsql/
-bash-3.2$ cp -rfp /var/lib/pgsql/backups/data_anterior_slave/*.conf /var/lib/pgsql/data/
```

Agora podemos continuar com as demais configurações da base STANDBY, removendo os archives dela pois quem faz esse trabalho é a MASTER:

```
-bash-3.2$ rm -rf /var/lib/pgsql/data/pg_xlog/*
-bash-3.2$ rm -rf /var/lib/pgsql/data/pg_archives
```

Agora vamos criar um arquivo de log para o pg_standby em um lugar estratégico (sugiro dentro da pasta pg_log conforme abaixo:

```
touch /var/lib/pgsql/data/pg_log/pg_standby.log
chown postgres:postgres /var/lib/pgsql/data/pg_log/pg_standby.log
```

Aqui tem mais um detalhe importante. O arquivo **/var/lib/pgsql/data/recovery.conf**.
Ele é o responsável por informar a base SLAVE que ela é uma SLAVE e que irá fazer nada mais nada menos que RESTORE com base nos archives gerados pela base MASTER. Dentro deste arquivo vai uma linha de comando do **pg_standby** que a base vai executar para ficar restaurando archives até segunda ordem[1].

Os comandos abaixo criam o arquivo/var/lib/pgsql/data/recovery.conf com o seu conteúdo que é o restore_command e acertam dono/grupo do arquivo:

```
echo "restore_command = 'pg_standby -t /var/lib/pgsql/pgsql.masterdown /var/lib/pgsql/master_archives %f %p %r
2>>/var/lib/pgsql/data/pg_log/pg_standby.log'" > /var/lib/pgsql/data/recovery.conf
chown postgres:postgres /var/lib/pgsql/data/recovery.conf
```

[1]Você deve estar se perguntando que segunda ordem é essa e que arquivo é esse /var/lib/pgsql/pgsql.masterdown na linha de comando do pg_standby. Trata-se de um arquivo trigger (ou gatilho) que o pg_standby reconhece como sendo um aviso pra ele se a base SLAVE continua sendo SLAVE e restaurando archives ou se ele pode encerrar a restauração e iniciar o banco em modo online normal aberto para conexões dos clientes (neste caso o SLAVE vira MASTER). Se o arquivo trigger existir, o ph_standby abre o banco SLAVE para uso como se fosse o master.. Este arquivo deve ser criado com muito cuidado pelo sistema de monitoramento do banco MASTER (geralmente heartbeat).

Explicacao de cada parametro na linha de comando do pg_standby:

```
-t: Arquivo trigger (serve para fazer failover)
/var/lib/pgsql/pgsql.masterdown é o próprio arquivo trigger
/var/lib/pgsql/master_archives é o diretório da maquina slave onde estão os archives gerados pelo master
```

```
%f %p %r é a sequência de archives
O resto do comando serve para jogar STDERR pra log em /var/lib/pgsql/data/pg_log/pg_standby.log.
```

Como neste caso essa base vai ser a SLAVE e nada mais, vamos remover este arquivo **por enquanto**.

```
rm -f /var/lib/pgsql/pgsql.masterdown
```

Podemos agora iniciar nosso banco de dados SLAVE:

```
[root@oracle02 ~]# /etc/init.d/postgresql restart
Stopping postgresql service:                               [  OK  ]
Starting postgresql service:                               [  OK  ]
```

Mensagens como essa dentro do log do pg_standby são normais segundo documentação oficial do PostgreSQL:

```
running restore                :cp: cannot stat `/var/lib/pgsql/master_archives/00000001.history': No such file or
directory
cp: cannot stat `/var/lib/pgsql/master_archives/00000001.history': No such file or directory
cp: cannot stat `/var/lib/pgsql/master_archives/00000001.history': No such file or directory
cp: cannot stat `/var/lib/pgsql/master_archives/00000001.history': No such file or directory
```

Neste momento, com os bancos MASTER e SLAVE no ar, se tudo correu bem conforme configurações acima, nosso esquema de replicação deve estar funcionando corretamente.

Agora um pouco sobre como podemos monitorar se a replicação está correndo bem e na sequência, como fazer um teste colocando a base SLAVE pra funcionar de verdade:

```
#=====================================
# PARA MONITORAR A BASE MASTER:
#=====================================
```

Podemos ver os últimos archives disponibilizados pela base MASTER no seguinte diretório:

```
-bash-3.2$ ls -la /var/lib/pgsql/data/pg_archives
```

É importante também ver se a data e hora dos últimos archives gerados condizem com a data e hora do presente momento. Por inúmeros motivos uma base Master pode parar de gerar archives e isso compromete nossa replicação.

Outra forma de controle é ver se o último archive disponibilizado no diretório citado acima é o mesmo que o processo do Archiver Process está informando:

```
-bash-3.2$ ps aux | grep -v grep | grep \"archiver process\"
```

```
#=====================================
# PARA MONITORAR A BASE SLAVE:
#=====================================
```

No próprio log do PostgreSQL podemos ver o status da aplicação dos archives na base SLAVE. Logo que iniciamos a operação da SLAVE, aparece algo assim no log:

```
-bash-3.2$ tail -f /var/lib/pgsql/data/pg_log/postgresql-`date '+%a'`.log
LOG:  creating missing WAL directory "pg_xlog/archive_status"
```

```
LOG:  starting archive recovery
LOG:  restore_command = 'pg_standby -t /var/lib/pgsql/pgsql.masterdown /var/lib/pgsql/master_archives %f %p %r
2>>/var/lib/pgsql/data/pg_log/pg_standby.log'
LOG:  restored log file "000000010000000300000A8.00000020.backup" from archive
LOG:  restored log file "000000010000000300000A8" from archive
LOG:  automatic recovery in progress
LOG:  redo starts at 3/A8000068, consistency will be reached at 3/A8000088
LOG:  consistent recovery state reached
LOG:  restored log file "000000010000000300000A9" from archive
LOG:  restored log file "000000010000000300000AA" from archive
LOG:  restored log file "000000010000000300000AB" from archive
LOG:  restored log file "000000010000000300000AC" from archive
```

E assim por diante o log deve ficar apresentando essas mensagens "restored log file from archive" indicando qual foi o arquivo que foi aplicado. Pela numeração podemos comparar os últimos archives que vão sendo aplicados em nossa base SLAVE, com os útlimos disponibilzados pela base MASTER (conforme comentado anteriormente neste artigo).

Para ver nosso "gatilho" funcionando, digo, para subir a base SLAVE como se fosse a MASTER, basta criar no servidor SLAVE o arquivo trigger que configuramos no pg_standby. Mas antes disso é legal criar uma situação que nos permita validar se um dado inserido agora no MASTER, realmente estará no SLAVE quando subirmos ele. Para isso, insira uma linha qualquer em uma tabela qualquer de um banco qualquer dentro de seu cluster PostgreSQL (Eu uso o pgAdmin III ou PhpPgAdmin pra esse tipo de manutenção). Em seguida forçamos a geração de um archivelog na base MASTER conforme procedimento explanado anteriormente neste artigo. Esperamos este archive ser aplicado em nossa base SLAVE (esperamos o "restored log file from archive" no log do banco).

Depois que temos certeza que o archive foi aplicado na SLAVE, podemos criar o gatilho:

```
-bash-3.2$ touch /var/lib/pgsql/pgsql.masterdown
```

Algo desse tipo deve aparecer no log do banco logo após a criação deste arquivo trigger:

```
-bash-3.2$ tail -0f /var/lib/pgsql/data/pg_log/postgresql-`date '+%a'`.log
LOG:  could not open file "pg_xlog/000000010000000400000008" (log file 4, segment 8): No such file or directory
LOG:  redo done at 4/7000068
LOG:  last completed transaction was at log time 2010-07-21 15:05:33.911492-03
LOG:  restored log file "000000010000000400000007" from archive
LOG:  selected new timeline ID: 2
LOG:  archive recovery complete
LOG:  autovacuum launcher started
LOG:  database system is ready to accept connections
```

Pronto, podemos consultar que as linhas que inserimos no banco MASTER agora pouco, estão disponíveis no banco SLAVE após aberto.
Lembrando que esse processo de subir a base SLAVE impossibilita fazer com que essa mesma base volte a ser uma base SLAVE. Uma vez que ela foi aberta para conexões, a única forma de ter uma nova base SLAVE é apagando-a e criando-a novamente passo-a-passo conforme fizemos acima.

Tenho um shellscript automatizado para controlar as bases MASTER e SLAVE em diversos aspectos comentados neste artigo. Quem tiver interesse basta entrar em contato comigo:

// Hudson Murilo dos Santos - Mobile: +55 (48) 9141-3303
// MSN: hudsantos@hotmail.com - Skype: hudsantos
// GoogleProfile: http://www.google.com/profiles/hudsantos
// LinkedIn: http://br.linkedin.com/pub/hudson-santos/16/181/412
// BLOG: http://www.cialinux.com.br
// CV: http://docs.google.com/View?id=d9zsdbk_224dw7bzwf9

Por: Hudson Murilo dos Santos
Referencias:

Não muito boas em minha opinião:

http://michsan.blogspot.com/2008/08/using-pgstandby-for-high-availability.html
http://www.psql.it/manuale/8.3/pgstandby.html

Oficiais (as melhores em minha opinião):

http://www.postgresql.org/docs/8.4/static/continuous-archiving.html
http://www.postgresql.org/docs/8.4/static/warm-standby.html
http://www.postgresql.org/docs/8.4/static/wal-configuration.html
http://www.postgresql.org/docs/8.4/static/pgstandby.html
http://www.postgresql.org/docs/8.4/static/high-availability.html

Recomendadas nas entrelinhas das oficiais:

http://anoncvs.postgresql.org/cvsweb.cgi/pgsql/contrib/pg_standby/
http://developer.postgresql.org/pgdocs/postgres/warm-standby.html

Funcoes administrativas: http://www.postgresql.org/docs/8.4/static/functions-admin.html