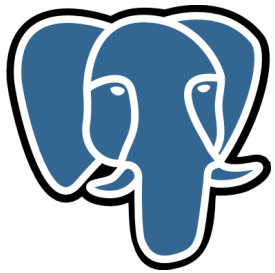
La replication dans PostgreSQL

Gilles Landais (CDS)

10 mars 2011



Sommaire

Introduction

Pourquoi répliquer

- sauvegarder une base dans un contexte de haute disponibilité
- load balancing
- (éventuellement) parallèliser les requêtes

Probèmatiques de la réplication

- disponiblité en lecture/ecriture *(maitre-escalve(s) ou maitre-maitre)*
- surcharge sur le maitre
- attente entre serveurs
- pertes en cas de failover
- granularité
- résolution de conflits

Les différents modes de réplications

- Réplication de système de fichiers

  DRBD sur Linux : sorte de RAID1 sur deux machines différentes

- Utilisation des fichiers journaux

  le "warm standby" et "hot standby " de PostgreSQL : utilisation des fichiers pg_xlog

- Réplication par trigger (ex : SLONY)
- Utilisation d'un middleware (ex : pgpool)
- réplication master/master en synchrone (Ex : Bucardo)

La réplication dans Postgresql9.0

Spécificité

- Réplication en mode asynchrone *(un maitre et un (ou +) esclave(s))*
- La base esclave peut devenir maitre suite à une intervention du DBA

Lorsque un esclave devient maitre ⇒ fin de la réplication. Pour remettre le serveur en esclave, il est nécésaire de le reconstruire.

Méthode interne de PostgreSQL, les fichiers WAL :

- Toutes les mises à jour de la base de données y sont tracées
- Permettent la reconstruction de la base en cas d'arret brutal
- recyclage des fichiers WAL

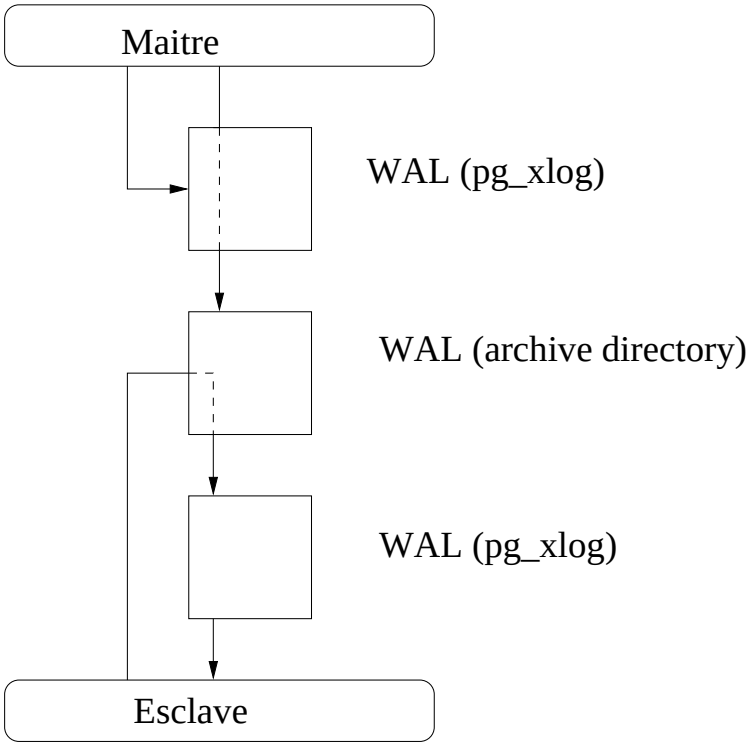
Les ordres de réplications sont séquentiels.
Une rupture de la chaine ⇒ attente inifinie de l'esclave

Modes de réplications

Warm standby : (version > 8.2 de PostgreSQL)

la base esclave n'est pas interogeable.

Principe de mise à jour :

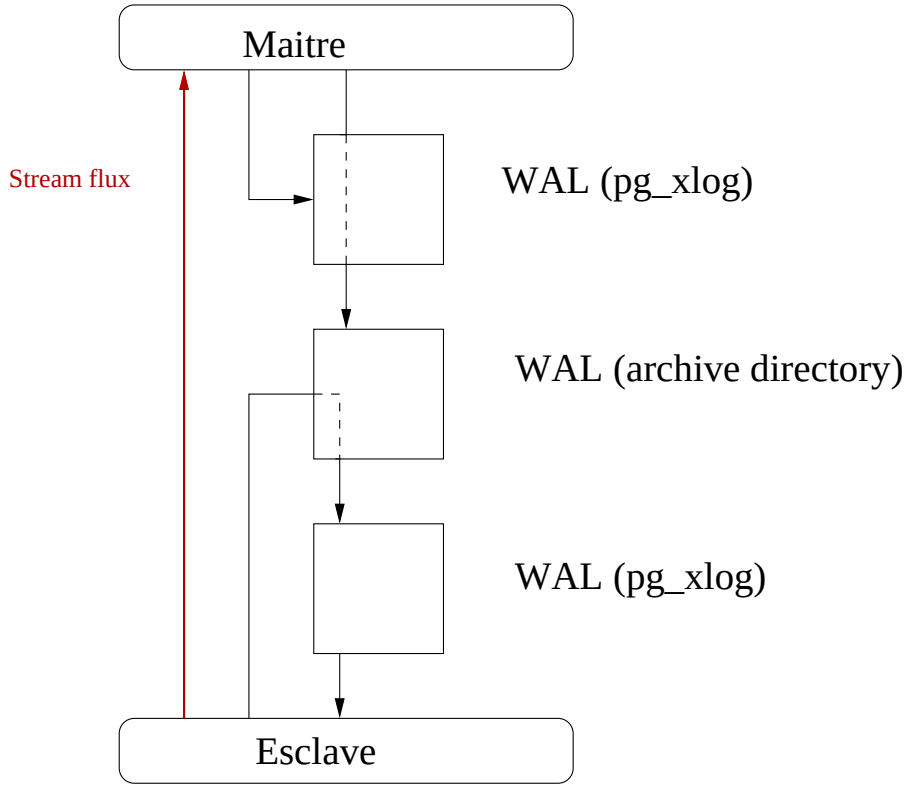


Hot standby : (version >9.0 de PostgreSQL)

Configurations possibles :

1. par copie des fichiers WAL (idem warm standby)
2. le mode streaming : l'esclave se connecte à la base maitre (plus réactif)

la base esclave est interogeable en mode read-only



Mise en place de la réplication en hot standby

Les serveurs hébergeant le maitre et l'esclave doivent être de même type

- Installation des outils de la « contrib »
- Préparation de la base esclave :
  1. initdb -D directory_base_esclave
  2. supprimer les fichiers et sous répertoires de la base esclave :

     global pg_xlog pg_tblspc pg_clog base
  3. sauvegarde de la base avec la commande

     select pg_start_backup()
  4. copie des même fichiers et sous répertoires de la base maitre sur l'esclave
  5. stopper le processus de sauvegarde du maitre

     select pg_stop_backup()
  6. mettre a jour la configuration de l'esclave

Par archivage des fichiers journaux (WAL)

Le serveur maitre

```
# postgresql.conf
wal_level = hot_standby # mettre 'archive' pour le warm standby
archive_mode = on
archive_command = 'cp -f %p archive_dir/%f </dev/null'
# Optionnel :
archive_timeout = 10 # force la creation de fichier WAL
```

Le serveur esclave

```
# postgresql.conf
hot_standby = on
max_standby_archive_delay = 30s # max delay before canceling queries (-1 = infini)
```

```
# recovery.conf
standby_mode = 'on'
restore_command = 'pg_standby -d -t /tmp/stopStandby archive_dir %f %p %r'
```

Le mode streaming

En mode streaming l'archivage des fichiers WAL est optionnel

Le serveur maitre

```
# postgresql.conf
wal_level = hot_standby
archive_mode = on
max_wal_senders=1 # >0 (nombre d'esclaves)
# Optionnel :
wal_sender_delay = 200ms# envoi des enregistrement a l'esclave
archive_command = 'cp -f %p archive_dir/%f </dev/null' # mais recommande
wal_keep_segments = 10 # si archive_command non utilise, permet de recycler plus de WAL
```

```
# pg_hba.conf
host replication postgres esclave_ip/32 trust
```

Le serveur esclave

```
# postgresql.conf
hot_standby = on
max_standby_archive_delay = 30s
max_standby_streaming_delay = 30s # max delay before canceling queries (coming from stream)
```

```
# recovery.conf
standby_mode = 'on'
primary_conninfo = 'host=maitre_ip port=5432 user=postgres password=foopass'
trigger_file = /tmp/stopStandby # met le serveur esclave en maitre
# Optionnel :
restore_command = 'cp archive_dir/%f %p'
archive_cleanup_command = 'pg_archivecleanup /path/to/archive %r'
```

Monitoring

- Surveiller le répertoire d'archivage s'il a été configuré
- Le point de synchronisation
  correspond au dernier fichier journal traité (global/pg_control)
- Il est possible a tout moment de connaitre ce point de synchronisation par la commande 'ps' :

```
postgres 24822 24821 0 19 :20 pts/1 00 :00 :00 /usr/local/pgsql/bin/postgres -D
/home/postgresql/9.0.2/base1
postgres 24825 24822 0 19 :20 ? 00 :00 :00 postgres : wal writer process
postgres 24827 24822 0 19 :20 ? 00 :00 :00 postgres : archiver process last was
000000010000000000000017
postgres 25050 25049 0 19 :28 pts/2 00 :00 :00 /usr/local/pgsql/bin/postgres -D
/home/postgresql/9.0.2/base2
postgres 25051 25050 0 19 :28 ? 00 :00 :00 postgres : startup process recovering
000000010000000000000017
postgres 25055 25050 0 19 :28 ? 00 :00 :00 postgres : writer process
```

Limitations et critiques du "hot standby"

- le mode read-only implique qu'il n'est pas possible d'utiliser les tables temporaires dans l'esclave
- des vérous posés sur l'esclave peuvent bloquer le mécanisme de réplication

  Le mécanisme de dévérouillage de ces vérous est basé sur un temps maximal d'exécution des requêtes sur l'esclave. (voir paramètres max_standby_archive_delay et max_standby_streaming_delay)
- blocage constaté lors d'un "drop" de table ou de "tablespace"
- le système de fichiers journaux peut poser des problèmes de volumétrie en cas d'arret du serveur esclave (si l'archivage est configuré)
- asynchrone : pertes possible de données en cas de failover

Avenir du hot_standby

- réplication synchrone dans la version 9.1
- bascule automatique maitre-esclave

Slony

Caractéristiques

- licence BSD - installation avec « aptitude » sur Ubuntu
- replication par trigger
- granularité au niveau table : principe d'abonnement
- démons mis a jour par des scripts Slony

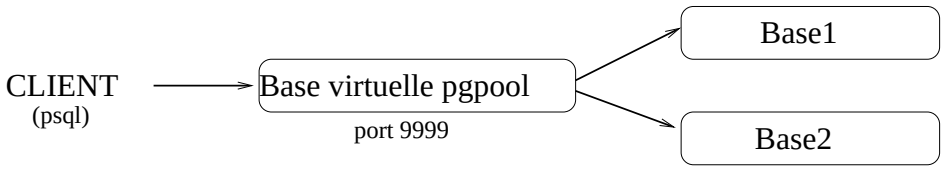
Avantages et limitations

- (+) possibilité d'utiliser des tables temporaires
- (+) plusieurs esclaves possibles
- (+) update, insert impossible sur le(s) esclave(s)
- (−) la réplication d'une table nécéssite une clef primaire
- (−) ajout de tables et modification du schema ⇒ script slony
- (−) moins rapide que hot_standby de postgres
- (−) asynchrone : pertes possible de données en cas de failover

pgpool-II

Caractéristiques

- licence BSD - installation avec « aptitude » sur Ubuntu
- middleware (SQL) : connection sur un port pgpool
- nombreuses fonctionnalités :
  pool de connections, caches, replications, load balancing, requêtes parallèles
- peut utiliser le mode streaming (hot_standby) de PostgreSQL ou Slony



pgpool-II

Configuration : load balancing + réplication
(2 serveurs PostgreSQL en mode normal)

- les requêtes de mises à jour sont exécutées sur les 2 serveurs
- bonne gestion des tables temporaires à condition de supprimer le cache de connection (ie : le pool)
- divergence possible en cas de connection directe
- en cas de divergence ⇒ passage en mode dégradé

Configuration : load balancing + Master Slave mode
(mode hot_standby de PostgreSQL)

- pas de divergence
- pgpool se charge d'exécuter les requêtes de mise à jour sur le maitre, et ce dans la même connection
- création de table temporaire ok, mais lecture impossible

Appendice

Comparaison des différents modes

Fonction	DRBD	WAL	Slony	pgpool-II	Bucardo
Plusieurs serveur maitres				ok	ok
Pas de surcharge sur le maitre		ok		ok	
Pas d'attente entre serveur		ok	ok		ok
Pas de perte en cas de panne du maitre	ok			ok	
Esclave disponible en lecture		Hot only	ok	ok	ok
Granularite			ok		ok
Pas de resolution de conflit	ok	ok	ok		