−Table des matières



# DRBD, la réplication des blocs disques

*Cet article, écrit par Guillaume Lelarge, a été publié dans le hors-série 45 du magazine GNU/Linux Magazine France, hors-série dédié aux administrateurs systèmes. Il est disponible maintenant sous licence Creative Commons.*

---


DRBD est un outil capable de répliquer le contenu d'un periphérique bloc. En ce sens, ce n'est pas un outil spécialisé pour PostgreSQL contrairement aux autres outils vus dans le hors-série 44 sur PostgreSQL. Il peut très bien servir à répliquer des serveurs de fichiers, ou de mails. Il réplique les données en temps réel et de façon transparente, pendant que les applications modifient leur fichiers sur un périphérique. Il peut fonctionner de façon synchrone ou asynchrone. Tout ça en fait donc un outil intéressant pour répliquer le répertoire des données de PostgreSQL.

Le cœur de DRBD est implanté sous la forme d'un module pour le noyau Linux. En ce sens, cet outil n'est pas du tout portable. En fait, il agit comme un pilote pour un périphérique bloc virtuel, ce qui le rend très flexible.

En dehors du module, DRBD dispose de trois outils d'administration utilisables en espace utilisateur. Nous n'allons travailler qu'avec drbdadm car il est une interface parfaite pour les deux autres outils (drbdsetup et drbdmeta).

Pour la démonstration, nous allons utiliser une partition d'un nouveau disque sur les deux serveurs. Cette partition sera montée sur le répertoire /var/lib/postgresql/8.4/main avant l'installation de PostgreSQL 8.4. Il faut néanmoins savoir que DRBD ne se limite pas à l'utilisation d'une partition. Vous pouvez lui donner à répliquer un périphérique RAID ou un volume LVM.

De plus, bien qu'il soit recommandé d'utiliser une connexion dédiée pour DRBD, nous n'allons pas le faire pour simplifier la démonstration.

Enfin, contrairement aux autres articles, nous supposerons ici que seule la distribution Debian est installée. Autrement dit, PostgreSQL n'est pas encore installé, et encore moins configuré.

## Installation

Il existe des paquets pour Debian 5.0. Ils ne correspondent pas à la dernière version de DRBD mais ils sont suffisamment récents pour nos besoins. Nous allons donc les utiliser. Cependant, en production, nous vous conseillons d'utiliser la dernière version, quitte à avoir à la compiler. Mais ici, un simple aptitude install doit suffire:

```
debian1:~# aptitude install drbd8-utils drbd8-modules-2.6.26-2-686
```

Ces deux paquets doivent aussi être installés sur debian2.

## Configuration

Le fichier de configuration se trouve directement dans le répertoire /etc. Plutôt que de le modifier, nous allons le renommer et le re-créer:

```
debian1:~# cd /etc
debian1:/etc# mv drbd.conf drbd.conf.distro
```

Voici le contenu du nouveau fichier drbd.conf:

```
global {
  usage-count yes;
}
common {
  protocol C;
}
resource postgresql {
  on debian1 {
    device    /dev/drbd1;
    disk      /dev/hdb1;
    address   192.168.10.66:7789;
    meta-disk internal;
  }
  on debian2 {
    device    /dev/drbd1;
    disk      /dev/hdb1;
    address   192.168.10.67:7789;
```

```
    meta-disk internal;
  }
}
```

La section globale n'existe qu'en un exemplaire. De tous les paramètres possibles, seul un sera utile à tout le monde. Usage-count permet au projet DRBD de collecter des statistiques d'utilisation des différentes versions de DRBD. Un serveur web est contacté à chaque fois qu'une nouvelle version de DRBD est installée sur un serveur. Notez que vous pouvez le désactiver en le configurant à 'no'. Cependant, nous vous conseillons de le laisser à 'yes'. Cette information est importante pour les développeurs de ce projet, c'est donc une contribution, minime mais respectable, de notre part, surtout que seul le numéro de version est envoyé.

La section common regroupe tous les paramètres communs aux différentes ressources. Nous n'indiquons ici que le protocole. Ce paramètre spécifie la façon dont la réplication est faite:

- protocole A pour une réplication asynchrone (les écritures locales sont considérées terminées une fois que l'écriture a eu lieu sur le disque local et que l'information a été enregistrée dans le tampon TCP d'envoi);
- protocole B pour une réplication synchrone en mémoire (les écritures locales sont considérées terminées une fois que l'écriture a eu lieu sur le disque local et que l'information a été reçue par le deuxième serveur);
- protocole C pour une réplication synchrone (les écritures locales sont considérées terminées une fois que l'écriture a eu lieu sur le disque local et sur le disque distant).

Le protocole A est sans aucun doute le plus rapide, mais aussi le moins sûr. Comme nous aimons nos données (sans quoi nous ne chercherions pas à nous protéger avec de la réplication), nous allons utiliser le protocole C.

Avec DRBD, un terme revient très fréquemment: la ressource. Une ressource est composée de plusieurs propriétés:

- Tout d'abord son nom qui sera utilisé pour la différencier des autres ressources.
- Le périphérique DRBD qui, une fois initialisé, pourra être formaté, monté et enfin utilisé.
- La configuration des disques associés.
- La configuration du réseau.

Il s'agit donc de la troisième partie du fichier de configuration. Y sont indiqués les différents nœuds à synchroniser et, pour chaque nœud, l'adresse IP, le périphérique disque réel, le périphérique disque virtuel de DRBD et la façon dont les méta-données sont enregistrées.

Le même fichier de configuration doit se trouver sur debian2.

## Mise en place de DRBD

La première chose à faire est de partitionner les deux nouveaux disques (le premier sur debian1, le second sur debian2).

```
debian1:/etc# fdisk /dev/hdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x4a30bcd4.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.


The number of cylinders for this disk is set to 4161.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-4161, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-4161, default 4161):
Using default value 4161

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

La même action doit être entreprise sur le serveur debian2.

Maintenant, ajoutons les métadonnées du périphérique du serveur debian1:

```
debian1:/etc# drbdadm create-md postgresql
Writing meta data...
initialising activity log
NOT initialized bitmap
New drbd meta data block sucessfully created.
success
```

Avant la suite, nous devons nous assurer que le module drbd est bien chargé:

```
debian1:/etc# lsmod | grep drbd
```

Comme il n'est pas chargé, nous nous en occupons:

```
debian1:/etc# modprobe drbd
```

Les trois commandes suivantes vont permettre d'associer la ressource DRBD à son périphérique, de configurer les paramètres de synchronisation pour cette ressource et enfin de se connecter:

```
debian1:/etc# drbdadm attach postgresql
debian1:/etc# drbdadm syncer postgresql
debian1:/etc# drbdadm connect postgresql
```

Tout est bon pour debian1. Voici ce que nous dit le fichier /proc/drbd :

```
debian1:/etc# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:WFConnection st:Secondary/Unknown ds:Inconsistent/DUnknown C r---
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
       resync: used:0/61 hits:0 misses:0 starving:0 dirty:0 changed:0
       act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Ce fichier permet de suivre l'état de la réplication. Ce qui nous intéresse surtout est la ligne « 1: ». cs indique le statut de la connexion, st l'état de la réplication et st le statut des disques. Et là, tout ça semble bon. Il n'y a pas de connexion car nous n'avons encore rien fait du côté de debian2. Nous allons donc y faire toutes les étapes précédentes:

```
debian2:/etc# drbdadm create-md postgresql
Writing meta data...
initialising activity log
NOT initialized bitmap
New drbd meta data block sucessfully created.
success
debian2:/etc# modprobe drbd
debian2:/etc# drbdadm attach postgresql
debian2:/etc# drbdadm syncer postgresql
debian2:/etc# drbdadm connect postgresql
/proc/drbd nous indique maintenant que la connexion est bien là:
debian1:/etc# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:Connected st:Secondary/Secondary ds:Inconsistent/Inconsistent C r---
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
       resync: used:0/61 hits:0 misses:0 starving:0 dirty:0 changed:0
       act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Il ne nous reste plus qu'à faire la synchronisation initiale avant de pouvoir utiliser le périphérique comme n'importe quel autre:

```
debian1:/etc# drbdadm -- --overwrite-data-of-peer primary postgresql
debian1:/etc# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:SyncSource st:Primary/Secondary ds:UpToDate/Inconsistent C r---
    ns:16896 nr:0 dw:0 dr:16896 al:0 bm:1 lo:0 pe:1 ua:0 ap:0
       [>....................] sync'ed:  1.0% (2080148/2097012)K
       finish: 0:57:46 speed: 416 (316) K/sec
       resync: used:0/61 hits:1053 misses:2 starving:0 dirty:0 changed:2
       act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Attention, les chiffres indiquées ne sont pas très représentatifs de ce que vous pouvez obtenir avec des disques réels. En effet, tous ces tests sont fait à partir d'une machine virtuelle (VirtualBox), d'où des performances disques minimales (pour rester poli…).

Il est tout à fait possible de continuer les préparatifs pendant la synchronisation, mais je préfère toujours attendre que cette partie soit terminée avant de continuer. Profitons-en pour regarder deux/trois fonctionnalités de drbdadm. Cet outil peut nous indiquer l'état d'une ressource, à savoir si le nœud où nous sommes est configuré en primaire ou en secondaire (autrement dit maître ou esclave):

```
debian1:/etc# drbdadm state postgresql
Primary/Secondary
```

debian1 est le nœud primaire/maître. Il est aussi possible de connaître l'état des disques quant à la réplication:

```
debian1:/etc# drbdadm dstate postgresql
UpToDate/Inconsistent
```

On (re-)découvre donc que le disque sur debian1 est à jour alors que celui sur debian2 est en cours de synchronisation initiale.

Une fois que la synchronisation est terminée, /proc/drbd contient quelque chose comme:

```
debian1:/etc# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r---
    ns:2097012 nr:0 dw:0 dr:2097012 al:0 bm:128 lo:0 pe:0 ua:0 ap:0
       resync: used:0/61 hits:130936 misses:128 starving:0 dirty:0 changed:128
       act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

et la commande d'état des disques affiche:

```
debian1:/etc# drbdadm dstate postgresql
UpToDate/UpToDate
```

Parfait, il faut maintenant formater et monter ce disque.

## Formatage et montage du disque

Le disque à considérer n'est plus /dev/hdb1 mais /dev/drbd1. Le formatage est assez classique:

```
debian1:/etc# mkfs.ext3 /dev/drbd1
mke2fs 1.41.3 (12-Oct-2008)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
Taille de fragment=4096 (log=2)
131072 i-noeuds, 524253 blocs
26212 blocs (5.00%) réservés pour le super utilisateur
Premier bloc de données=0
Nombre maximum de blocs du système de fichiers=536870912
16 groupes de blocs
32768 blocs par groupe, 32768 fragments par groupe
8192 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        32768, 98304, 163840, 229376, 294912

Écriture des tables d'i-noeuds : complété
Création du journal (8192 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété

Le système de fichiers sera automatiquement vérifié tous les 34 montages ou
après 180 jours, selon la première éventualité. Utiliser tune2fs -c ou -i
pour écraser la valeur.
```

Il faut ensuite passer au montage. Le but est d'y placer le répertoire des données de PostgreSQL. Il existe donc deux possibilités: soit le monter dans /var/lib/postgresql et il n'y aura rien de plus à faire, soit le monter sur un répertoire complètement différent et il faudra faire un initdb (pg_createcluster sous Debian) vers le bon répertoire. Nous allons choisir la première méthode, celle qui nous génère le moins de travail:

```
debian1:/etc# mkdir /var/lib/postgresql
debian1:/etc# mount /dev/drbd1 /var/lib/postgresql
```

## Installation de PostgreSQL sur debian1

Il ne nous reste plus qu'à installer PostgreSQL. Avoir avoir configuré le fichier /etc/apt/sources.list comme indiqué dans l'article sur l'installation dans le hors-série 44, il ne reste plus qu'à faire l'habituel:

```
debian1:/etc# aptitude update
[... message de progression ...]
debian1:/etc# aptitude install postgresql-8.4
[... message de progression ...]
```

Ceci terminé, nous avons le nouveau répertoire des données dans /var/lib/postgresql:

```
debian1:/etc# ll /var/lib/postgresql/8.4/main/
total 48
drwx------ 5 postgres postgres 4096 aoû 29 18:26 base
drwx------ 2 postgres postgres 4096 aoû 29 18:26 global
drwx------ 2 postgres postgres 4096 aoû 29 18:25 pg_clog
drwx------ 4 postgres postgres 4096 aoû 29 18:25 pg_multixact
drwx------ 2 postgres postgres 4096 aoû 29 18:26 pg_stat_tmp
drwx------ 2 postgres postgres 4096 aoû 29 18:25 pg_subtrans
drwx------ 2 postgres postgres 4096 aoû 29 18:25 pg_tblspc
drwx------ 2 postgres postgres 4096 aoû 29 18:25 pg_twophase
-rw------- 1 postgres postgres    4 aoû 29 18:25 PG_VERSION
drwx------ 3 postgres postgres 4096 aoû 29 18:25 pg_xlog
-rw------- 1 postgres postgres  133 aoû 29 18:26 postmaster.opts
-rw------- 1 postgres postgres   54 aoû 29 18:26 postmaster.pid
```

## Copie des fichiers de configuration de debian1 sur debian2

Comme nous ne synchronisons « que » le répertoire /var/lib/postgresql et que Debian déplace les fichiers de configuration dans /etc/postgresql, il nous faut manuellement copier les fichiers de configuration sur le serveur esclave.

```
debian1:/etc# scp -r /etc/postgresql debian2:/etc
root@debian2's password:
start.conf                                    100%  378     0.4KB/s   00:00
environment                                   100%  316     0.3KB/s   00:00
pg_ctl.conf                                   100%  143     0.1KB/s   00:00
postgresql.conf                               100%   16KB  16.5KB/s   00:00
pg_hba.conf                                   100% 3822     3.7KB/s   00:00
pg_ident.conf                                 100% 1631     1.6KB/s   00:00
```

## Pour le redémarrage du serveur

Pour que PostgreSQL puisse démarrer, il faut que le disque soit monté, donc il faut aussi que DRBD soit lancé. Le script de lancement de DRBD doit être configuré pour démarrer avant celui de PostgreSQL. Or, dans certains cas, ce n'est pas fait ainsi. Par exemple, sous Debian, il va être nécessaire de modifier le nom du lien symbolique. Nous allons utiliser pour cela l'exécutable update-rc.d:

```
debian1:/etc# rm rc*.d/*drbd
debian1:/etc# update-rc.d drbd defaults 18
 Adding system startup for /etc/init.d/drbd ...
   /etc/rc0.d/K18drbd -> ../init.d/drbd
   /etc/rc1.d/K18drbd -> ../init.d/drbd
   /etc/rc6.d/K18drbd -> ../init.d/drbd
   /etc/rc2.d/S18drbd -> ../init.d/drbd
   /etc/rc3.d/S18drbd -> ../init.d/drbd
   /etc/rc4.d/S18drbd -> ../init.d/drbd
   /etc/rc5.d/S18drbd -> ../init.d/drbd
```

## Utilisation de PostgreSQL

L'utilisation de PostgreSQL ne change en rien. Vous pouvez créer des objets, faire des insertions, des modifications et des mises à jour de données. Vous pouvez aussi les consulter. Bref, une utilisation identique… avec certainement des performances moindres. En effet, chaque modification doit se faire sur les deux machines en même temps. L'aspect synchrone est à ce prix.

Voici quelques modifications effectuées dans le cadre de cette démonstration:

```
postgres@debian1:~$ createdb b1
postgres@debian1:~$ psql b1
psql (8.4.0)
Saisissez « help » pour l'aide.

b1=# CREATE TABLE t1 (id integer);
CREATE TABLE
b1=# INSERT INTO t1 SELECT i FROM generate_series(1, 10000) AS i;
INSERT 0 10000
b1=# INSERT INTO t1 SELECT i FROM generate_series(1, 10000) AS i;
INSERT 0 10000
b1=# \q
postgres@debian1:~$ createdb b2
psql b2
postgres@debian1:~$ psql b2
psql (8.4.0)
Saisissez « help » pour l'aide.

b2=# CREATE TABLE t2 (id integer, autre text);
CREATE TABLE
b2=# INSERT INTO t2 SELECT i, 'Ligne '||i FROM generate_series(1, 100000) AS i;
INSERT 0 100000
b2=# CREATE TABLE t3 (id integer, autre text, autre2 date);
CREATE TABLE
b2=# INSERT INTO t3 SELECT i, 'Ligne '||i, now() FROM generate_series(1, 100000) AS i;
INSERT 0 100000
b2=# DELETE FROM t2 WHERE id BETWEEN 40000 AND 50000;
DELETE 10001
b2=# UPDATE t3 SET autre=upper(autre) WHERE id BETWEEN 20000 AND 25000;
UPDATE 5001
b2=# \q
```

## Testons le switchover

Réaliser un switchover prend un peu de temps à cause de toutes les pelures à enlever avant d'arriver au niveau de DRBD. Il faut tout d'abord enlever la « pelure » PostgreSQL:

```
debian1:/etc# /etc/init.d/postgresql-8.4 stop
Stopping PostgreSQL 8.4 database server: main.
```

Ensuite, il faut enlever la « pelure » disque virtuel en démontant le disque:

```
debian1:/etc# umount /dev/drbd1
```

À ce moment-là, DRBD est toujours dans le même état:

```
debian1:/etc# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r---
    ns:192336 nr:0 dw:192336 dr:161 al:49 bm:0 lo:0 pe:0 ua:0 ap:0
       resync: used:0/61 hits:0 misses:0 starving:0 dirty:0 changed:0
       act_log: used:0/127 hits:48035 misses:63 starving:0 dirty:14 changed:49
```

Par contre, comme les deux pelures sont enlevées, on va pouvoir déclasser debian1 en serveur secondaire:

```
debian1:/etc# drbdadm secondary postgresql
debian1:/etc# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:Connected st:Secondary/Secondary ds:UpToDate/UpToDate C r---
    ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
       resync: used:0/61 hits:0 misses:0 starving:0 dirty:0 changed:0
       act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Et enfin, on peut reclasser debian2 en serveur primaire:

```
debian2:~# drbdadm primary postgresql
debian2:~# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r---
    ns:0 nr:192336 dw:192336 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
       resync: used:0/61 hits:0 misses:0 starving:0 dirty:0 changed:0
       act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Maintenant que debian2 est primaire au niveau DRBD, il faut remettre en place les pelures. On commence par monter le disque:

```
debian2:~# mkdir /var/lib/postgresql
debian2:~# mount /dev/drbd1 /var/lib/postgresql
debian2:~# ll /var/lib/postgresql/8.4/main/
total 44
drwx------ 7 104 106 4096 sep 27 17:36 base
drwx------ 2 104 106 4096 sep 27 17:38 global
drwx------ 2 104 106 4096 sep 27 17:23 pg_clog
drwx------ 4 104 106 4096 sep 27 17:23 pg_multixact
drwx------ 2 104 106 4096 sep 27 17:38 pg_stat_tmp
drwx------ 2 104 106 4096 sep 27 17:23 pg_subtrans
drwx------ 2 104 106 4096 sep 27 17:23 pg_tblspc
drwx------ 2 104 106 4096 sep 27 17:23 pg_twophase
-rw------- 1 104 106    4 sep 27 17:23 PG_VERSION
drwx------ 3 104 106 4096 sep 27 17:36 pg_xlog
-rw------- 1 104 106  133 sep 27 17:23 postmaster.opts
```

Avant de démarrer PostgreSQL, il faut l'installer. /var/lib/postgresql/8.4/main et /etc/postgresql/8.4/main existant déjà, le paquet ne va pas lancer de nouveau initdb. Détectant la présence d'un cluster complet, il lance PostgreSQL à partir des données que nous avons déjà grâce à la réplication de DRBD. Par contre, avant de lancer PostgreSQL, il nous faut aussi vérifier que les fichiers de configuration soit lisible par l'utilisateur postgres, et pour cela nous devons créer l'utilisateur postgres:

```
debian2:/etc# useradd postgres
debian2:/etc# chown -R postgres:postgres /etc/postgresql/8.4/main
debian2:/etc# chown -R postgres:postgres /var/lib/postgresql/8.4/main
debian2:/etc# vim /etc/apt/sources.list
debian2:/etc# aptitude update
debian2:/etc# aptitude install postgresql-8.4
[... messages de progression ...]
Paramétrage de postgresql-8.4 (8.4.0-2~bpo50+1) ...
Starting PostgreSQL 8.4 database server: main.
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Lecture de l'information d'état étendu
Initialisation de l'état des paquets... Fait
Écriture de l'information d'état étendu... Fait
Lecture des descriptions de tâches... Fait
```

Testons maintenant le contenu du nouveau serveur primaire:

```
debian2:/etc# su - postgres
postgres@debian2:~$ psql -l
                             Liste des bases de données
   Nom    | Propriétaire | Encodage |     Tri     | Type caract. |    Droits d'accès
----------+--------------+----------+-------------+--------------+-----------------------
 b1       | postgres     | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  |
 b2       | postgres     | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  |
 postgres | postgres     | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  |
 template0 | postgres    | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  | =c/postgres
                                                                 : postgres=CTc/postgres
 template1 | postgres    | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  | =c/postgres
                                                                 : postgres=CTc/postgres
(5 lignes)

postgres@debian2:~$ psql b1
psql (8.4.0)
Saisissez « help » pour l'aide.

b1=# \d
        Liste des relations
 Schéma | Nom | Type  | Propriétaire
--------+-----+-------+--------------
 public | t1  | table | postgres
(1 ligne)

b1=# SELECT count(*) FROM t1;
 count
-------
 20000
(1 ligne)

b1=# \q
postgres@debian2:~$ psql b2
psql (8.4.0)
Saisissez « help » pour l'aide.

b2=# \d
        Liste des relations
```

```
 Schéma | Nom | Type  | Propriétaire
--------+-----+-------+--------------
 public | t2  | table | postgres
 public | t3  | table | postgres
(2 lignes)

b2=# SELECT count(*) FROM t2;
 count
-------
 89999
(1 ligne)

b2=# SELECT count(*) FROM t3;
 count
--------
 100000
(1 ligne)
```

Faisons quelques modifications avant de tester le failover:

```
postgres@debian2:/$ createdb b3
postgres@debian2:/$ psql b1
psql (8.4.1)
Saisissez « help » pour l'aide.

b1=# INSERT INTO t1 SELECT i FROM generate_series(20000, 40000) AS i;
INSERT 0 20001
b1=# \q
```

## Et maintenant, le failover

Vu que nous avons des données, nous allons pouvoir tester le failover. Simulons une panne sur le nœud primaire (l'arrêter suffit). Voici ce que nous avons sur le nœud secondaire (debian1 actuellement):

```
debian1:/etc# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:WFConnection st:Secondary/Unknown ds:UpToDate/DUnknown C r---
    ns:0 nr:18260 dw:18260 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
       resync: used:0/61 hits:0 misses:0 starving:0 dirty:0 changed:0
       act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Il a bien détecté la déconnexion (ou l'absence) de debian2. L'état du nœud 2 est inconnu, mais le nœud 1 est à jour. Mettons maintenant le nœud 1 en nœud primaire:

```
debian1:/etc# drbdadm primary postgresql
debian1:/etc# cat /proc/drbd
version: 8.0.14 (api:86/proto:86)
GIT-hash: bb447522fc9a87d0069b7e14f0234911ebdab0f7 build by phil@fat-tyre, 2008-11-12 16:40:33

 1: cs:WFConnection st:Primary/Unknown ds:UpToDate/DUnknown C r---
    ns:0 nr:18260 dw:18260 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
       resync: used:0/61 hits:0 misses:0 starving:0 dirty:0 changed:0
       act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

debian1 est bien devenu le nœud primaire. Nous pouvons enfin monter le disque:

```
debian2:/etc# mount /dev/drbd1 /var/lib/postgresql
debian2:/etc# ll /var/lib/postgresql/8.4/main/
base/          pg_clog/       pg_stat_tmp/   pg_tblspc/     PG_VERSION     postmaster.opts
global/        pg_multixact/  pg_subtrans/   pg_twophase/   pg_xlog/
Tout a l'air d'être là. Il ne nous reste plus qu'à démarrer PostgreSQL:
debian1:/etc# chown -R postgres:postgres /var/lib/postgresql/8.4/main
debian1:/etc# /etc/init.d/postgresql-8.4 start
Starting PostgreSQL 8.4 database server: main.
```

Testons si la nouvelle base de données b3 existe bien:

```
debian1:/etc# su - postgres
postgres@debian1:~$ psql -l
                            Liste des bases de données
   Nom    | Propriétaire | Encodage |     Tri     | Type caract. |    Droits d'accès
----------+--------------+----------+-------------+--------------+-----------------------
 b1       | postgres     | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  |
 b2       | postgres     | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  |
 b3       | postgres     | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  |
 postgres | postgres     | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  |
 template0 | postgres    | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  | =c/postgres
                                                                 : postgres=CTc/postgres
 template1 | postgres    | UTF8     | fr_FR.UTF-8 | fr_FR.UTF-8  | =c/postgres
                                                                 : postgres=CTc/postgres
(6 lignes)
```

C'est bien le cas. Nos dernières modifications sur debian2 sont bien là.

## Petit récapitulatif

### Avantages majeurs

- plutôt simple à mettre en place et à configurer
- réplication synchrone
- documentation excellente
- intégration à Heartbeat / Pacemaker très facile

### Inconvénients majeurs

- pas de granularité sur les objets à répliquer
- esclaves inaccessibles
- lenteur à prévoir
- deux serveurs uniquement (une configuration trois nœuds est disponible à partir de la version 8.3 de DRBD)

### Inconvénient « mineur »

- fonctionne sous Linux uniquement

## Conclusion

DRBD est un système simple à mettre en place. Son gros avantage est la possibilité d'avoir une réplication synchrone, son inconvénient direct est sa lenteur.