Sandhose

Les sauvegardes de Zeste de Savoir

20 janvier 2019

Table des matières

% LES SAUVEGARDES DE ZESTE DE SAVOIR % Sandhose % 11 mars 2018

1. Introduction

J'ai dans mon appart un petit serveur tournant sous FreeBSD, me servant notamment de routeur et de NAS. J'ai 8 To de stockage dessus en RAID[1], et me sert depuis peu de ce serveur comme espace de sauvegarde pour la production de Zeste de Savoir.

Les données de Zeste de Savoir se trouvent à deux endroits en production. D'une part dans le dossier `/opt/zds/data`, contenant les dépôts git des contenus, les médias uploadés, et autres PDFs générés. D'autre part dans une base de donnée MySQL.

2. Sauvegarde de la base via Xtrabackup

Auparavant, pour sauvegarder la base, `mysqldump` était utilisé pour générer un dump complet de la base sous forme de fichier `sql`. Ces dumps pèsent un peu moins de 500 Mo non-compressés, et 100 Mo `gzip`pés. Aussi, chaque dump contient l'ensemble de la base de données, alors qu'en général entre deux sauvegardes, seul une partie de la base a changé.

Aussi, `mysqldump` a tendance à vouloir verrouiller les tables pendant le dump, et ainsi ralentir le site.

Mon objectif était d'effectuer des backups plus régulièrement, pour éviter des retours dans le passés trop importants en cas d'incident majeur. Les dumps étaient effectués une fois tous les jours, et je ne pouvais pas me permettre d'en faire un toutes les heures.

Xtrabackup est un outil développé par Percona permettant de faire des sauvegardes incrémentales et sans `LOCK` d'une base MySQL. Il se base sur le fait qu'InnoDB[2] maintient sur le disque un historique des transactions lui permettant de récupérer la base en cas d'incident majeur, tant que le disque est intacte. Il utilise donc directement les fichiers InnoDB sur le disque, plutôt que de récupérer toute la base via une connexion mysql au serveur de BDD.

Il offre donc

1. J'ai en réalité ~16 To de disques durs répartis sur 4 disques, qui sont en miroirs deux par deux.

— des sauvegardes sans lock des tables[3]
— des sauvegardes incrémentales, puisqu'il lui suffit de sauvegarder les nouvelles entrées du journal de transactions d'InnoDB
— une restauration beaucoup plus rapide, puisqu'il s'agit d'une simple copie de fichier, et non un énorme script SQL à exécuter

---

J'ai donc écrit un petit script permettant de faire des backups complètes et incrémentales. Les backups complètes sont effectuées une fois par jours, et les backups incrémentales toutes les heures.

```
1  ##!/bin/sh
2
3  ## Bonne pratique: arrêter le script si une commande retourne un
       code d'erreur autre que 0,
4  # et si une variable non-existante est utilisée
5  set -eu
6
7  ## L'endroit où sont stockées les sauvegardes
8  WD=/var/backups/mysql
9  ## `latest` est un lien symbolique vers la dernière sauvegarde
10 LATEST=$WD/latest
11
12 ## On récupère le nom complet de l'ancienne backup
13 PREVIOUS=`readlink -f $LATEST`
14 ## La nouvelle backup aura comme nom AAMMJJ-HHSS
15 NEXT=$WD/`date '+%Y%m%d-%H%M'`
16
17 if [ -d "$NEXT" ]; then
18     echo "\`$NEXT' already exists."
19     exit 1
20 fi
21
22 if [ "$#" -ge 1 ] && [ "$1" = "full" ]; then
23     # Les backups "full" ont en plus `-full` dans le nom
24     NEXT=$NEXT-full
25     xtrabackup --backup --target-dir=$NEXT 2> $NEXT.log
26 else
27     if ! [ -L "$LATEST" ]; then
28         echo
               "\`$LATEST' does not exists. Consider doing a full backup first."
29         exit 1
30     fi
31
32     xtrabackup --backup --target-dir=$NEXT
           --incremental-basedir=$PREVIOUS 2> $NEXT.log
33 fi
34
35 ## On supprime les anciens liens symboliques
```

```
rm -f $LATEST $LATEST.log

## Et on les recréé avec la sauvegarde fraîche
ln -s $NEXT $LATEST
ln -s $NEXT.log $LATEST.log
```

Plus qu'à exécuter ce script régulièrement le script via cron :

```
root@zdsprod1:~# crontab -l
## min hour dom month dow command
0 * * * * /var/backups/mysql/backup.sh
15 3 * * * /var/backups/mysql/backup.sh full
```

Et voilà le résultat !

```
root@zdsprod1:~# ls -l /var/backups/mysql
…
drwxr-x---  5 root root 4.0K Mar 10 02:00 20180310-0200
-rw-r--r--  1 root root  47K Mar 10 02:00 20180310-0200.log
drwxr-x---  5 root root 4.0K Mar 10 03:00 20180310-0300
-rw-r--r--  1 root root  47K Mar 10 03:00 20180310-0300.log
drwxr-x---  5 root root 4.0K Mar 10 03:15 20180310-0315-full
-rw-r--r--  1 root root  46K Mar 10 03:15 20180310-0315-full.log
drwxr-x---  5 root root 4.0K Mar 10 04:00 20180310-0400
-rw-r--r--  1 root root  46K Mar 10 04:00 20180310-0400.log
drwxr-x---  5 root root 4.0K Mar 10 05:00 20180310-0500
-rw-r--r--  1 root root  46K Mar 10 05:00 20180310-0500.log
…
```

À titre indicatif, une sauvegarde complète pèse 1.2 Go non-compressée, 200 Mo compressé, et une incrémentale pèse en moyenne 15 Mo.

Ces sauvegardes sont ensuite récupérées sur mon serveur toutes les heures via `rsync` :

```
rsync -avr zdsprod1:/var/backups/mysql/
    /tupperware/backups/zestedesavoir/db
```

Histoire de pas exploser le stockage de la prod, j'ai écrit un petit script pour garder que les deux dernières backups complètes (+ les incrémentales entre). Les anciennes sauvegardes restent bien sûr sur mon serveur.

```
##!/bin/sh

set -eu

WD=/var/backups/mysql

BACKUPS="`echo $WD/*-*/ | tr ' ' '\n' | sort -nr`"

TO_DELETE="`
    echo "$BACKUPS" | awk '
        BEGIN { full=0 }
        { if (full > 1) { print $0 } }
        /full/ { full++ }
    '
`"

[ -z "$TO_DELETE" ] || rm -r "$TO_DELETE"
```

3. rsync et snapshots ZFS

J'ai donc mentionné plus tôt que j'utilise ZFS sur mon serveur comme système de fichier.

ZFS a été développé à l'origine par Oracle pour Solaris. Il a été ensuite porté notamment sur FreeBSD, et plus récemment sur Linux (via ZFSOnLinux).

ZFS a la particularité de faire à la fois gestionnaire de volumes logiques (comme LVM) & RAID (mdadm sur linux) et du système de fichier. Il permet de gérer un ensemble de disques durs comme un seul ensemble (éventuellement avec de la redondance pour être tolérant aux pannes), que l'on peut subdiviser en multiples sous-volumes (`datasets`, dans le jargon ZFS) avec différentes propriétés. On pourra donc attribuer différents droits aux utilisateurs en fonction du volume, limiter la taille maximum de chaque volume, et activer certaines fonctionnalités type compression et deduplication à la volée par volume.

ZFS permet également de faire des snapshots instantanément et sans coût d'un volume. Une snapshot ne va prendre comme place sur le disque que la différence entre la snapshot et le volume actif. Idéal donc, pour des backups.

```
spaetzle/root ~ > zfs list -t snap
NAME                                                    USED  AVAIL
    REFER  MOUNTPOINT
…
```

2. Le moteur de stockage par défaut sur MySQL
3. Je sais, `mysqldump` peut ne pas locker les tables, mais il lock quand même les transactions pendant la backup, ce qui "met en pause" toute opération pendant le dump.

```
tupperware/backups/zestedesavoir/data@20180310-1602   412K      -
    5.56G  -
tupperware/backups/zestedesavoir/data@20180310-1702   884K      -
    5.56G  -
tupperware/backups/zestedesavoir/data@20180310-1802   876K      -
    5.56G  -
tupperware/backups/zestedesavoir/data@20180310-1902   844K      -
    5.57G  -
tupperware/backups/zestedesavoir/data@20180310-2303      0      -
    5.57G  -
tupperware/backups/zestedesavoir/data@20180311-0002      0      -
    5.57G  -
tupperware/backups/zestedesavoir/data@20180311-0102      0      -
    5.57G  -
…
```

Les snapshots sont donc très légères, avec en général moins d'1 Mo par snapshot.

Le script pour synchroniser le dossier `/opt/zds/data` et les backups de la base, et faire la snapshot est relativement simple :

```
##!/bin/sh

BASE=tupperware/backups/zestedesavoir

echo "Syncing data"
rsync -azvr --delete zdsprod1.zestedesavoir.com:/opt/zds/data/
    /$BASE/data
echo "Creating snapshot"
zfs snapshot $BASE/data@`date '+%Y%m%d-%H%M'`
echo "Syncing database"
rsync -azvr zdsprod1.zestedesavoir.com:/var/backups/mysql/
    /$BASE/db
```

Ce script est lancé 1 fois par heure, 2 min après la backup incrémentale de la base.

4. La cerise sur le gâteau : compression à la volée des volumes

ZFS permet de compresser des volumes entiers de manière transparente.

```
spaetzle/root ~ > zfs get used,compressratio,compression,logicalused tupperware/backups/zestedesavoir/{db,data}
NAME                                      PROPERTY       VALUE          SOURCE
tupperware/backups/zestedesavoir/data     used           5.57G          -
tupperware/backups/zestedesavoir/data     compressratio  1.17x          -
tupperware/backups/zestedesavoir/data     compression    lz4            local
tupperware/backups/zestedesavoir/data     logicalused    5.89G          -
tupperware/backups/zestedesavoir/db       used           666M           -
tupperware/backups/zestedesavoir/db       compressratio  3.11x          -
tupperware/backups/zestedesavoir/db       compression    lz4            local
tupperware/backups/zestedesavoir/db       logicalused    1.88G          -
```

Ainsi, si les backups de la base semblent peser près de 2 Go, sur le disque ils ne prennent que 700 Mo.

Ça fait partie des fonctionnalités qui rendent ZFS *awesome*. Sachez que je n'ai couvert ici qu'une petite partie des fonctionnalités de ZFS.

5. Conclusion

Conclusion, j'aime ZFS.

À plus ou moins long terme j'aimerais mettre en place un autre serveur à un autre endroit, pour notamment avoir de la redondance à deux endroits physiques distincts sur mes backups. Oh, est-ce que j'ai mentionné que ZFS permet d'envoyer et recevoir des datasets entiers ou incrémentalement via de simples pipes ?

Et les backups de ZdS sont à nouveau un peu plus sérieuse, et on devrait pouvoir réagir plus rapidement en cas d'incident.