Sauvegarde incrémentielle avec rsync

Table of Contents

Cet article va décrire un moyen de créer une solution de sauvegarde incrémentielle simple en utilisant **RSYNC** et des liens physiques.

RSYNC est un outil de ligne de commande couramment utilisé sur Linux et d'autres systèmes d'exploitation de type UNIX pour copier et synchroniser les répertoires.

Par défaut, **RSYNC** ne compresse pas les fichiers. La restauration des fichiers est aussi simple qu'une commande CP.

A propos des fichiers et des inodes

Avant d'entrer dans les détails, il faut comprendre comment les fichiers sont stockés sur le système de fichiers et le fonctionnement des liens physiques.

Tous les fichiers et répertoires sont représentés dans le système de fichiers par un numéro Inode qui est l'identité interne du système de fichiers pour le fichier.

Pour voir les numéros d'inode répertoriés exécuter `ls -li dans un répertoire`

Un «fichier» tel qu'on peut le voir par chemin et nom de fichier est en fait une référence à l'inode et est souvent appelé «lien». Lorsqu'on créer un lien physique d'un fichier à un autre, on créer une référence (lien) distincte d'un nouveau nom de fichier au même numéro Inode. Ceci est différent d'un lien «symbolique» (SymLink) qui est une référence d'un emplacement à un autre chemin du système de fichiers:

- Lorsqu'on modifie le fichier d'origine, les modifications sont également visibles dans la version physique
- Lorsqu'on modifie le fichier lié, les modifications sont également visibles dans le fichier d'origine
- La modification de la propriété et des autorisations affecte également les deux fichiers
- Si on supprime le fichier d'origine, on verra que le lien physique existe toujours et que le contenu du fichier reste intact, on peut même créer un autre lien physique et supprimer celui existant et les données restent intactes (en fait lorsqu'on supprime un fichier à l'aide de la commande RM ou de toute autre méthode, cela consiste simplement à supprimer le lien vers l'inode. C'est pourquoi la fonction pour supprimer un fichier dans des langues telles que C et

PHP est appelée «Unlink». Lorsque tous les liens vers un inode ont été supprimés, l'inode lui-même sera supprimé. Tant qu'il y a au moins un lien pointant vers lui l'inode et les données resteront intactes). En revanche, un lien symbolique pointant vers le fichier d'origine ne sera plus valide

Par exemple pour créer un miroir d'un répertoire distant `/data` d'un serveur nommé `server1` dans un répertoire local `/backup/server1`. Typiquement, on fait quelque chose comme ceci :

```
rsync -av --delete server1:/data/ /backup/server1/
```

On exécute ensuite la même commande à chaque fois qu'on veut mettre à jour le miroir avec les dernières modifications du serveur.

Pour implémenter un système de sauvegarde incrémentielle de base, on fait une copie locale de la sauvegarde précédente avant de démarrer le **rsync**:

```
cp -a /backup/server1/ /backup/server1Old/
```

Ensuite, on met à jour le miroir depuis le serveur distant :

```
rsync -av --delete server1:/data/ /backup/server1/
```

Évidemment, ce n'est pas très efficace pour le temps ou l'espace, on peut améliorer cela en utilisant les liens physiques.

Sauvegarde incrémentielle avec cp

On peut créer des liens physiques en ajoutant l'argument **-l** à la commande **cp**:

```
# Créer un clone lié en dur de la sauvegarde actuelle
cp -al /backup/server1 /backup/server1Old
# mettre à jour le miroir depuis le serveur distant
rsync -av --delete server1:/data/ /backup/server1/
```

Cette sauvegarde est conservée dans `/backup/server1Old` et `/backup/server1` contiendra l'intégralité de la nouvelle sauvegarde et n'utilisera que l'espace requis pour les fichiers nouveaux et modifiés. Cela crée un moyen efficace d'implémenter des sauvegardes incrémentielles, mais cela à ses limites, en particulier lorsqu'il s'agit de traiter un grand nombre de fichiers.

Sauvegarde incrémentielle avec rsync

Pour améliorer encore les choses, la fonctionnalité **–link-dest** de **rsync** permet de créer efficacement des copies liées en dur du contenu d'un répertoire, seuls les fichiers modifiés occuperont de l'espace sur le disque.

Périmètre

En prenant ceci comme point de départ :

- `server1:/data`: répertoire source distant
- `/backup/server1New`: destination d'une nouvelle sauvegarde. N'existe pas encore
- `/backup/server1Old`: sauvegarde précédente existante

Le résultat qu'on veut dans `/backup/server1New` est que tous les fichiers inchangés sont des liens physiques vers les fichiers existants dans `/backup/server1Old` et seuls les fichiers modifiés sont copiés depuis le serveur distant et occupent de l'espace dans la nouvelle sauvegarde.

Algorithme link-dest

La première fois que RSYNC est exécutée, la destination est créée et la source complète est copiée sur destination. Par la suite, seuls les changements de source sont copiés sur destination. Lorsque l'option **–link-dest** est utilisée, les fichiers inchangés sont liés en dur à la sauvegarde précédente.

Un lien physique est un pointeur vers un fichier. Les liens physiques ont l'avantage d'utiliser très peu de mémoire.

Voici comment l'algorithme “RSync –Link-Dest = Dir” crée des fichiers dans la destination:

```
    Si la destination n'existe pas,
        créer une destination

    Si Dir existe (où DIR est la sauvegarde précédente),
        Comparer la source à Dir
        Faire un lien physique à DIR pour les fichiers inchangés
        Copier les fichiers modifiés à partir de la source
    autre
        Copier tous les fichiers de la source
```

Utilisation

Pour l'utiliser,il suffit simplement d'ajouter comme argument **–link-dest “ancien répertoire”** à la commande **rsync**:

```
rsync -av --link-dest /backup/server1Old server1:/data/ /backup/server1New/
```

rsync effectue une sauvegarde normale de `server1:/data` vers `/backup/server1New` mais si le fichier n'existe pas dans `/backup/server1New`, il examinera le même chemin relatif sous `/backup/server1Old` pour voir si le fichier a changé. Si le fichier dans `/backup/server1Old` est le même que le fichier sur le serveur distant, au lieu de le copier, il créera un lien physique du fichier dans `/backup/server1Old` vers `/backup/server1New`.

Analyse de la sauvegarde

On regarde le contenu de l'ancien répertoire de sauvegarde :

```
[user1@backupbox ~]$ ls -lRi /backup/server1Old/
/backup/server1Old/:
total 0
68876 drwxrwxr-x. 3 user1 user1 53 Oct  2 17:30 files

/backup/server1Old/files:
total 72
33651935 drwxrwxr-x. 2 user1 user1    42 Oct  2 17:30 bar
   68882 -rw-rw-r--. 1 user1 user1 28883 Oct  2 17:30 foo1
   68883 -rw-rw-r--. 1 user1 user1 27763 Oct  2 17:30 foo2
   68884 -rw-rw-r--. 1 user1 user1 10487 Oct  2 17:30 foo3

/backup/server1Old/files/bar:
total 76
33695759 -rw-rw-r--. 1 user1 user1 32603 Oct  2 17:30 bar1
33838984 -rw-rw-r--. 1 user1 user1 15318 Oct  2 17:30 bar2
33839003 -rw-rw-r--. 1 user1 user1 26122 Oct  2 17:30 bar3
```

Sur le serveur on modifie alors un fichier :

```
echo "Hello world" >/home/data/files/foo3
```

Puis, on exécute la commande de sauvegarde incrémentielle :

```
rsync -av --link-dest=/backup/server1Old server1:/home/data/
/backup/server1New/
receiving incremental file list
created directory /backup/server1New
files/foo3

sent 136 bytes  received 272 bytes  816.00 bytes/sec
total size is 130,701  speedup is 320.35
```

On voit dans la sortie de rsync que seul le fichier modifié a été copié, mais si on liste le contenu du nouveau répertoire, on peut voir qu'il contient tous les fichiers :

```
[user1@backupbox ~]$ ls -lRi /backup/server1New/
/backup/server1New/:
total 0
101051460 drwxrwxr-x. 3 user1 user1 53 Oct  2 17:30 files

/backup/server1New/files:
total 64
```

```
    68885 drwxrwxr-x. 2 user1 user1    42 Oct  2 17:30 bar
    68882 -rw-rw-r--. 2 user1 user1 28883 Oct  2 17:30 foo1
    68883 -rw-rw-r--. 2 user1 user1 27763 Oct  2 17:30 foo2
101051461 -rw-rw-r--. 1 user1 user1    12 Oct  2 17:40 foo3

/backup/server1New/files/bar:
total 76
33695759 -rw-rw-r--. 2 user1 user1 32603 Oct  2 17:30 bar1
33838984 -rw-rw-r--. 2 user1 user1 15318 Oct  2 17:30 bar2
33839003 -rw-rw-r--. 2 user1 user1 26122 Oct  2 17:30 bar3
```

Si on compare les numéros d'inode à ceux de la liste de `/backup/server1Old` ci-dessus, on voit que seuls le fichier modifié et les répertoires ont des numéros d'inode différents.

En utilisant du, on pouvons également voir que la deuxième sauvegarde prend moins d'espace sur le disque :

```
[user1@backupbox ~]$ du -chs /backup/server1*
140K    /backup/server1New
12K /backup/server1Old
152K    total
```

Scripte de sauvegarde incrémentielle

Voici un exemple de script qui peut être utilisé pour créer des sauvegardes incrémentielles quotidiennes d'un répertoire. Chaque sauvegarde est stockée dans un répertoire nommé d'après la date du jour et il recherchera la sauvegarde d'hier pour créer les liens physiques :

```
#!/bin/bash

set -o errexit
set -o nounset
set -o pipefail

# Où stocker les sauvegardes incrémentielles
DESTBASE=/backup/nom_serveur
# Le chemin source à sauvegarder. Peut être local ou distant.
readonly SOURCE_DIR="/data/petra/"
# Répertoire de la sauvegarde du jour
readonly BACKUP_DIR="$DESTBASE/$(date +%Y-%m-%d)"
# Répertoire de la sauvegarde précédente
readonly LATEST_LINK=""$DESTBASE/$(date -d yesterday +%Y-%m-%d)/"

# Utiliser la sauvegarde d'hier comme base incrémentielle si elle existe
if [ -d "$LATEST_LINK" ]
then
OPTS="--link-dest $LATEST_LINK"
fi
```

```
# Exécute le rsync
rsync -av --delete $OPTS "$SOURCE_DIR" "$BACKUP_DIR"
```

De cette façon chaque sauvegarde quotidienne est un miroir complet du répertoire distant. Cela signifie qu'aucune logique complexe n'est requise pour trouver la dernière version d'un fichier ou pour trouver un fichier à partir d'une date spécifique, il suffit d'accéder simplement au répertoire nommé avec la date souhaitée et ouvrir le fichier normalement.

Chaque répertoire de sauvegarde est complètement indépendant des autres, donc si on a besoin de libérer de l'espace, on peut simplement supprimer les sauvegardes dont on n'a plus besoin. La suppression d'une sauvegarde n'aura pas d'impact sur les sauvegardes avant ou après, un simple **rm -rf** est tout ce dont on av besoin !

Limites

Comme pour toute solution de sauvegarde, celle-ci a ses limites et il faut choisir une méthode qui correspond le mieux aux besoins d'utilisation particulier. Voici quelques exemples de limitations de cette solution :

- Les changements d'autorisations ou de propriété sur un fichier source signifient que le fichier est compté comme un nouveau fichier, il sera donc copié à nouveau même si son contenu n'a pas changé. Il existe des options dans **rsync** pour contrôler ce comportement.
- Si on déplace ou renomme un fichier sur le serveur source, il comptera comme un nouveau fichier et sera entièrement copié même si son contenu n'a pas changé et qu'il a toujours le même numéro d'inode.
- Les répertoires eux-mêmes ne peuvent pas être liés en dur sur la plupart des systèmes de fichiers, cela n'est donc pas pris en charge par **rsync**. Dans la plupart des cas d'utilisation, ce n'est pas un problème, mais si on a un nombre énorme de répertoires dans la sauvegarde, ils commenceront à prendre une quantité notable d'espace sur le disque de sauvegarde.