BTRFS, LE SYSTÈME DE FICHIERS NOUVELLE GÉNÉRATION

par Carl Chenet,
[Ingénieur système GNU/Linux, contributeur Debian et Pythoniste]

Le système de fichiers btrfs a gagné en fonctionnalité et en stabilité au fil des dernières versions du noyau Linux. Appelé à remplacer ext3 et ext4 dans un avenir de plus en plus proche, cumulant les fonctionnalités d'un système de fichiers moderne et la flexibilité d'un gestionnaire de volumes logiques comme LVM, btrfs suscite un intérêt croissant pour ceux qui souffraient du manque de flexibilité des solutions actuelles. Sachant que le stockage a été amené à subir de très importantes évolutions ces dernières années, les nouvelles fonctionnalités de btrfs sont donc très attendues.

Nous présenterons dans cet article les fonctionnalités importantes de btrfs que les administrateurs de système GNU/Linux et les utilisateurs avancés utiliseront bientôt au quotidien.

1 Quelques outils pour btrfs

Pour présenter btrfs, nous utilisons la distribution GNU/Linux Debian Wheezy avec le noyau Linux 3.2.0. En effet, btrfs évolue rapidement, l'utiliser sur un noyau plus vieux peut exposer à un manque de fonctionnalités ou à des bugs aujourd'hui corrigés avec les noyaux plus récents.

Installons quelques outils qui nous seront utiles pour jouer avec nos partitions et nos systèmes de fichiers btrfs dans la suite de cet article :

```
# apt-get install btrfs-tools parted
```

2 Migration de ext3 vers btrfs

Tout d'abord, la base : créer une nouvelle partition sur notre système grâce à laquelle on va pouvoir jouer avec btrfs. Pour cela, nous utilisons l'utilitaire **fdisk** de la façon suivante en tant qu'utilisateur **root** :

```
root@btrfs:~# fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00071371

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    19531775     9764864   83  Linux
/dev/sda2        19531776    21485567      976896   82  Linux swap / Solaris
```

Notre système ne contient pour l'instant que deux partitions, à savoir la partition principale avec tout notre système et une partition de swap. Créons immédiatement une nouvelle partition :

```
# fdisk /dev/sda

Command (m for help): n
Partition type:
   p   primary (2 primary, 0 extended, 2 free)
   e   extended
Select (default p): p
Partition number (1-4, default 3):
Using default value 3
First sector (21485568-41943039, default 21485568):
Using default value 21485568
Last sector, +sectors or +size{K,M,G} (21485568-41943039, default 41943039): +2G

Command (m for help): p

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00071371

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    19531775     9764864   83  Linux
/dev/sda2        19531776    21485567      976896   82  Linux swap / Solaris
/dev/sda3        21485568    25679871     2097152   83  Linux

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

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Périphérique ou
ressource occupé.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
# partprobe
```

On crée un bon vieux système de fichiers ext3 sur la nouvelle partition :

```
# mkfs.ext3 /dev/sda3
```

Oups, mais j'ai tout faux là, on veut parler de btfs et non d'ext3 que tout administrateur système connaît parfaitement. Pas de panique, un outil existe et vous permettra de migrer simplement vos systèmes de fichiers ext3 existants vers btrfs.

La commande **btrfs-convert** va nous permettre de convertir notre système de fichiers en ext3 vers le format btrfs le plus simplement du monde :

```
# btrfs-convert /dev/sda3
creating btrfs metadata.
creating ext2fs image file.
cleaning up system chunk.
conversion complete.
```

Nous montons maintenant notre système de fichiers btrfs de la façon la plus habituelle possible à l'aide de la commande **mount** :

```
# mkdir /mybtrfs1
# mount -t btrfs /dev/sda3 /mybtrfs1/
```

Nous avons créé un répertoire **mybtrfs1** à la racine du système dans lequel nous avons monté notre système de fichiers btrfs. On vérifie que le montage est bien pris en compte sur notre système à l'aide du fichier **/etc/mtab** :

```
# grep btrfs /etc/mtab
/dev/sda3 /mybtrfs1 btrfs rw,relatime,space_cache 0 0
```

La dernière ligne retournée par la commande précédente indique bien un système de fichiers de type btrfs monté en lecture/écriture (**rw**).

3 Créer un système de fichiers btrfs à partir d'une partition vide

Il est bien évidemment possible de créer un système de fichiers btrfs à partir d'une partition vide. Nous reprenons la partition **/dev/sda3** créée au chapitre précédent et créons un système de fichiers btrfs grâce à la commande suivante :

```
# mkfs.btrfs /dev/sda3

WARNING! - Btrfs Btrfs v0.19 IS EXPERIMENTAL
WARNING! - see http://btrfs.wiki.kernel.org before using

fs created label (null) on /dev/sda3
        nodesize 4096 leafsize 4096 sectorsize 4096 size 2.00GB
Btrfs Btrfs v0.19
# mount /dev/sda3 /mybtrfs1/
```

À l'aide de l'utilitaire **mkfs.btrfs**, nous créons un système de fichiers btrfs sur notre partition, puis nous la montons dans le répertoire **/mybtrfs**.

```
# grep btrfs /etc/mtab
/dev/sda3 /mybtrfs1 btrfs rw,relatime,space_cache 0 0
```

Afin de s'assurer que la partition a été correctement montée, nous consultons la liste des systèmes de fichiers montés présente dans le fichier **/etc/mtab**.

4 Modifier la taille d'un système de fichiers btrfs

Sur un système utilisant plusieurs partitions ou lorsque vous prévoyez une évolution dans le temps de la taille d'une partition, le support d'opérations de modification à chaud de la taille offert par le système de fichiers utilisé est primordial. Nous donnons dans la suite de ce chapitre un exemple complet des opérations offertes par btrfs.

Dans un premier temps, nous consultons l'état de notre système de fichiers avant de commencer nos opérations :

```
# df -h |grep sda3
/dev/sda3                2,0G  504K  1,86   1% /mybtrfs1
```

Notre système de fichiers fait bien 2 gigaoctets. Pour rendre cet exemple intéressant, nous allons générer un fichier d'une taille assez grosse et nous assurer qu'il n'est pas modifié au cours de nos manipulations du système de fichiers sous-jacent.

```
# dd if=/dev/zero of=/mybtrfs1/bigfile bs=1G count=1
1+0 enregistrements lus
1+0 enregistrements écrits
1073741824 octets (1,1 GB) copiés, 61,2058 s, 17,5 MB/s
```

Nous créons ici un fichier d'un gigaoctet à l'aide de la commande **dd**.

```
# md5sum /mybtrfs1/bigfile
cd573cfaace07e7949bc0c46028904ff  /mybtrfs1/bigfile
```

Puis nous générons et notons la somme de hachage MD5 de ce fichier, qui va nous servir de référence pour déterminer si nos opérations sur le système de fichiers ont eu un impact sur nos données.

4.1 Réduire la taille du système de fichiers

Nous commençons par vouloir réduire la taille de notre système de fichiers btrfs :

```
# btrfs filesystem resize -512M /mybtrfs1/
Resize '/mybtrfs1/' of '-512M'
```

Nous procédons ici au redimensionnement à chaud de notre système de fichiers btrfs en lui enlevant de l'espace, une opération peu conseillée sur la plupart des systèmes de fichiers.

```
# df -h |grep sda3
/dev/sda3                1,5G  1,1G  279M  79% /mybtrfs1
```

Nous contrôlons maintenant la taille de notre système de fichiers. Nous constatons qu'elle a bien diminué.

```
# md5sum /mybtrfs1/bigfile
cd573cfaace07e7949bc0c46028904ff  /mybtrfs1/bigfile
```

Le contrôle de la somme de hachage de notre fichier nous montre que nos données n'ont pas été affectées par le récent changement.

4.2 Augmenter la taille du système de fichiers

Passons maintenant à l'opération suivante, à savoir faire croître la taille de notre système de fichiers :

```
# btrfs filesystem resize +512M /mybtrfs1/
Resize '/mybtrfs1/' of '+512M'
```

L'opération se déroule correctement et la commande nous retourne un résumé de l'action effectuée.

```
# md5sum /mybtrfs1/bigfile
cd573cfaace07e7949bc0c46028904ff  /mybtrfs1/bigfile
```

De nouveau un contrôle de nos données qui s'avère concluant.

```
# df -h |grep sda3
/dev/sda3                2,0G  1,1G  791M  57% /mybtrfs1
```

Notre système de fichiers a bien retrouvé sa taille originale.

Après nous être intéressés à la réduction et à l'agrandissement d'un système de fichiers btrfs, observons le cas limite, à savoir demander à faire décroître la taille du volume jusqu'à une taille inférieure à celle du total des données présente dans le volume. Dans notre exemple d'un volume de 2 gigaoctets avec un fichier présent de 1 gigaoctet, nous allons tenter de décroître notre volume jusqu'à 500 mégaoctets :

```
# btrfs filesystem resize -1500M /mybtrfs1/
Resize '/mybtrfs1/' of '-1500M'
ERROR: unable to resize '/mybtrfs1/' - No space left on device
```

La commande **btrfs** nous confirme l'impossibilité d'exécuter cet ordre pour conserver la cohérence du volume. Elle nous retourne également un code d'erreur particulier (30 dans ce cas précis).

```
# df -h |grep sda3
/dev/sda3                2,0G  1,1G  791M  57% /mybtrfs1
```

Et en effet, nous avons la confirmation que la taille du volume n'a pas été modifiée.

4.3 Quand la place vient à manquer dans votre système de fichiers

L'un des grands intérêts de LVM a été de permettre d'augmenter la taille d'un système de fichiers existant en ajoutant une nouvelle partition LVM provenant généralement d'un nouveau support physique. Le même type de fonctionnalité est offert par btrfs, encore une fois sans avoir plus à se préoccuper de redimensionner le système de fichiers présent au-dessus du gestionnaire de volumes logiques.

Dans le cas où nous souhaitons ajouter de l'espace à notre partition btrfs existante, nous nous assurons dans un premier temps d'avoir deux partitions btrfs disponibles. Nous utilisons la commande suivante pour obtenir un état des volumes btrfs présent sur notre système :

```
# btrfs filesystem show
Label: none  uuid: a3d87604-84e7-4459-b671-ce2189562d16
        Total devices 1 FS bytes used 28.00KB
        devid    1 size 2.00GB used 240.75MB path /dev/sda4

Label: none  uuid: 0524b3e8-fae1-4afc-b3b4-aea37d8e4103
        Total devices 1 FS bytes used 424.00KB
        devid    1 size 2.00GB used 2.00GB path /dev/sda3
# df -h |grep mybtrfs1
/dev/sda3                2,0G  464K  1,8G   1% /mybtrfs1
```

/dev/sda3 et **/dev/sda4** sont donc des systèmes de fichiers btrfs utilisables et **/dev/sda3** est déjà monté dans le répertoire **/mybtrfs1**.

Un simple ajout de la partition inutilisée dans le système de fichiers monté suffit :

```
# btrfs device add /dev/sda4 /mybtrfs1/
```

Nous vérifions immédiatement le bon déroulement de l'opération et constatons que 2 gigaoctets ont bien été ajoutés à la taille totale et à la taille disponible du système de fichiers :

```
# df -h |grep mybtrfs1
/dev/sda3                4,0G  400K  3,8G   1% /mybtrfs1
```

Enfin, si vous souhaitez répartir les données et répliquer les métadonnées du système de fichiers sur la ou les nouvelles partitions btrfs récemment ajoutées, il vous suffit d'employer la commande suivante :

```
# btrfs filesystem balance /mybtrfs1/
# btrfs filesystem df /mybtrfs1/
Data, RAID0: total=3.51GB, used=128.00KB
System, RAID1: total=8.00MB, used=4.00KB
System: total=4.00MB, used=0.00
Metadata, RAID1: total=204.75MB, used=28.00KB
```

Les informations retournées par le **df** de btrfs nous indique clairement la situation : les données (Data) sont en RAID0 entre les différentes partitions constituant le système de fichiers alors que le système et les méta-données sont eux en RAID1. Cela signifie que même si vous perdez l'une des partitions du système, vous serez toujours capable de monter la partition restante. Par contre, les données n'étant pas répliquées mais réparties, vous aurez perdu celles présentes sur la partition disparue.

5 Compression à la volée des données dans un volume btrfs

Le système de fichiers btrfs offre une intéressante fonctionnalité permettant de compresser à la volée les données que vous enregistrez dans un volume btrfs. Cette option peut s'avérér très utile pour des volumes que vous consacrez par exemple au stockage sur vos serveurs de production. Illustrons par un exemple la mise en place de cette solution avec notre volume btrs monté dans **/mybtrfs1**.

Commençons par constater l'utilisation de l'espace sur notre système de fichiers btrfs :

```
# btrfs filesystem df /mybtrfs1
Data: total=1.41GB, used=1008.32MB
System, DUP: total=8.00MB, used=4.00KB
System: total=4.00MB, used=0.00
Metadata, DUP: total=102.38MB, used=1.43MB
Metadata: total=8.00MB, used=0.00
```

L'espace utilisé par nos données sur le volume est de 1008 mégaoctets. Pouvons-nous gagner de la place sur cet espace ?

Nous commençons par effacer le fichier puis par démonter le volume :

```
# rm /mybtrfs/bigfile && umount /mybtrfs1
```

Puis nous le montons de nouveau en passant l'option suivante :

```
# mount -t btrfs -o compress=lzo /dev/sda3 /mybtrfs1/
```

À l'aide de la commande **dd**, nous créons de nouveau un fichier de taille équivalente contenant les mêmes données :

```
# dd if=/dev/zero of=/mybtrfs1/bigfile bs=1G count=1
```

Puis nous constatons l'espace utilisé :

```
# btrfs filesystem df /mybtrfs1
Data: total=1.41GB, used=986.55MB
System, DUP: total=8.00MB, used=4.00KB
System: total=4.00MB, used=0.00
Metadata, DUP: total=102.38MB, used=1.55MB
Metadata: total=8.00MB, used=0.00
```

Nous avons gagné de l'espace. Le gain peut être beaucoup plus conséquent sur des arborescences de fichiers réels, de tailles diverses.

Pour s'en convaincre, nous allons renouveler l'opération, mais cette fois en ajoutant l'option **compress-force** de btrfs lorsque l'on monte le volume. Cette option permet de forcer la compression sur des fichiers que btrfs compresse mal par défaut d'habitude, ce qui est le cas pour des fichiers de taille importante créés à l'aide de la commande **dd**. En cause, le comportement de l'algorithme de compression à la volée de btrfs qui essaie d'épargner le processeur lorsqu'il détermine selon ses premières opérations qu'un fichier peut difficilement être compressé **[1]** :

```
# mount -o compress=zlib,compress-force /dev/sda3 /mybtrfs1/
# dd if=/dev/zero of=/mybtrfs1/bigfile bs=1G count=1
1+0 enregistrements lus
1+0 enregistrements écrits
1073741824 octets (1,1 GB) copiés, 39,1517 s, 27,4 MB/s
# btrfs filesystem df /mybtrfs1/
Data: total=1.41GB, used=32.94MB
System, DUP: total=8.00MB, used=4.00KB
System: total=4.00MB, used=0.00
Metadata, DUP: total=102.38MB, used=1.53MB
Metadata: total=8.00MB, used=0.00
```

Tout de suite le taux de compression est beaucoup plus sérieux, la taille des données sur notre volume btrfs n'étant plus que de 32.94 mégaoctets.

Afin de pratiquer la même opération d'activation de la compression directement depuis un système de fichiers btrfs monté, nous pouvons utiliser la commande suivante qui activera l'option de compression et compressera les données déjà présentes :

```
# btrfs filesystem defragment -czlib /mybtrfs1
```

Utiliser l'option de compression prend tout son sens pour des volumes restreints en taille ou pour économiser de la place de façon automatique, par exemple sur le stockage de fichiers générés par des utilisateurs - qui compressent rarement eux-mêmes leurs données - et peu appelés à être modifiés dans le temps.

6 Mettre en place un RAID1 à l'aide de btrfs

Le système de fichiers btrfs vous permet de mettre en place simplement une solution logicielle de RAID composée de différents volumes. Actuellement, les types 0, 1 et 10 sont supportés. Nous développerons dans l'exemple suivant la mise en place d'un RAID 1 (réplication des mêmes données sur deux volumes). Précisons rapidement que s'il est tout à fait possible de créer un RAID1 composé de deux volumes présents physiquement sur un même disque, il est de bon sens de constituer un RAID1 avec deux volumes étant chacun sur un disque physique différent, pour ne perdre qu'une seule des partitions en cas de panne matérielle.

Pour mettre en place notre RAID1, nous commençons par l'initialiser de la façon suivante :

```
# mkfs.btrfs -m raid1 -d raid1 /dev/sda3 /dev/sda4

WARNING! - Btrfs Btrfs v0.19 IS EXPERIMENTAL
WARNING! - see http://btrfs.wiki.kernel.org before using

adding device /dev/sda4 id 2
fs created label (null) on /dev/sda3
        nodesize 4096 leafsize 4096 sectorsize 4096 size 4.00GB
Btrfs Btrfs v0.19
```

Un RAID1 entre les deux volumes a été créé. Nous pouvons vérifier cela assez facilement en écrivant un fichier sur l'un des deux volumes du RAID1 monté puis en vérifiant s'il a été répliqué sur la seconde branche du RAID1.

```
# mount /dev/sda3 /mybtrfs1/
# echo "this is a test" > /mybtrfs1/test
```

Nous montons la première branche du RAID1 et écrivons un fichier dedans.

```
# umount /mybtrfs1/
# mount /dev/sda4 /mybtrfs1/
# ls /mybtrfs1/
test
# cat /mybtrfs1/test
this is a test
```

Nous démontons ensuite la première branche puis montons la seconde branche que nous n'avions pas encore manipulée jusque-là. Nous constatons, grâce à la commande **ls**, qu'un fichier est bien présent et contient bien la chaîne de caractères attendue.

Constatons la bonne configuration en miroir de nos deux partitions btrfs à l'aide de la commande suivante :

```
# btrfs filesystem df /mybtrfs1/
Data, RAID1: total=204.75MB, used=128.00KB
Data: total=8.00MB, used=0.00
System, RAID1: total=8.00MB, used=4.00KB
System: total=4.00MB, used=0.00
Metadata, RAID1: total=204.75MB, used=24.00KB
Metadata: total=8.00MB, used=0.00
```

La fonctionnalité de RAID offerte par btrfs est très simple à mettre en place lorsque vous souhaitez utiliser du RAID logiciel sur votre système. Ce dernier présente de nombreux avantages intéressants face au RAID matériel, parmi lesquels celui d'être standard, d'utiliser des outils présents sur toutes les distributions et de ne pas confier vos données à la discrétion d'une boîte noire que représentent les cartes matérielles propriétaires pour lesquelles bien souvent corruption du miroir rime avec perte définitive de vos données.

7 Les sous-volumes

Le sous-volume est un composant de base de l'organisation hiérarchique des éléments dans btrfs. Nous allons le présenter ici car il est nécessaire de bien comprendre son fonctionnement étant donné qu'une fonctionnalité essentielle de btrfs, à savoir les instantanés (*snapshots*), repose sur les sous-volumes.

Toujours en nous basant sur notre système de fichiers btrfs créé à partir de la partition **/dev/sda3** et monté dans **/mybtrfs1**, nous allons maintenant créer puis manipuler un sous-volume.

```
# btrfs subvolume create /mybtrfs1/subvolume1
Create subvolume '/mybtrfs1/subvolume1'
```

Nous commençons par créer le sous-volume. Précisons que l'existence d'un sous-volume est matérialisée au niveau de l'arborescence par un répertoire présent à la racine du point de montage du volume.

Cherchons à obtenir quelques informations sur ce dernier :

```
# btrfs subvolume list /mybtrfs1/
ID 266 top level 5 path subvolume1
```

Le numéro **266** identifie de manière unique notre sous-volume. Le chemin **subvolume1** est également indiqué. Un élément important est ici à prendre en compte : le chemin **subvolume1**, qui est aussi le nom du sous-volume, est relatif à la racine de montage de notre volume btrfs.

```
# ll /mybtrfs1/
total 0
drwx------ 1 root root 0 mars  11 21:38 subvolume1
```

Nous créons maintenant un fichier dans notre volume btrfs :

```
# touch /mybtrfs1/root-volume
```

Puis nous démontons le système de fichiers btrfs racine pour remonter son sous-volume au même emplacement :

```
# umount /mybtrfs1
# mount -o subvol=subvolume1 /dev/sda3 /mybtrfs1/
```

Listons maintenant le contenu de **/mybtrfs1** :

```
# ll /mybtrfs1/
total 0
```

Le fichier **/mybtrfs1/root-volume** n'apparaît pas. Ce qui est normal étant donné que volume et sous-volume ne partagent pas leurs données.

Vous pouvez également monter votre sous-volume à partir de son identifiant :

```
# mount -o subvolid=266 /dev/sda3 /mybtrfs1
# grep mybtrfs /etc/mtab
/dev/sda3 /mybtrfs1 btrfs rw,relatime,space_cache 0 0
```

8 Les instantanés (snapshots)

Après avoir présenté les sous-volumes, il est assez simple d'aborder la notion d'instantané.

Il s'agit simplement d'un sous-volume particulier capable de partager ses données avec un autre sous-volume. Les instantanés sont très intéressants car ils autorisent un retour arrière vers l'état initial du volume avant la création de l'instantané. Les utilisateurs emploieront ce retour arrière lorsqu'ils auront lourdement modifié une arborescence de fichiers au point de ne plus savoir où ils en sont ou s'ils ont perdu des données.

On peut également envisager l'emploi de nombreux instantanés permettant de suivre l'évolution d'un système de fichiers dans le temps, par exemple un instantané par jour sur un mois afin de pouvoir retrouver l'état de son système de fichiers sur un mois.

8.1 Création de l'instantané et retour arrière

Prenons un exemple concret pendant lequel nous allons monter notre système de fichiers btrfs, déposer un fichier dans l'arborescence du volume courant puis créer un instantané. Nous démonterons ensuite le volume courant pour monter l'instantané puis modifier l'arborescence en y ajoutant un fichier supplémentaire. Effectuant enfin notre retour arrière, nous démonterons l'instantané pour remonter le volume d'origine, ce dernier ne contenant que le premier fichier créé. Présentons les détails pas-à-pas :

Nous monterons notre système de fichiers btrfs et créons un fichier :

```
# mount /dev/sda3 /mybtrfs1
# echo "Powered by Debian" > /mybtrfs/foo1
```

Nous créons maintenant depuis le volume courant l'instantané qui va nous permettre de faire ce qui nous passe par la tête sur cette arborescence de fichiers :

```
# btrfs subvolume snapshot /mybtrfs1 /mybtrfs1/snapshot1
```

La commande précédente indique que nous souhaitons créer un sous-volume de type instantané du volume monté dans **/mybtrfs1** et qui s'appelera **/mybtrfs1/snapshot1**. Comme attendu, un répertoire est apparu à la racine de notre arborescence portant le nom de l'instantané.

```
# ll /mybtrfs1/
total 8
-rw-r--r-- 1 root root 18 mars  15 23:03 foo1
dr-xr-xr-x 1 root root 34 mars  15 23:14 snapshot1
```

Nous démontons maintenant le volume courant pour monter à la place notre instantané dans lequel nous créons un nouveau fichier :

```
# mount -o subvol=snapshot1 /dev/sda3 /mybtrfs1/
# echo "Debian: When it's ready" > /mybtrfs1/foo2
```

Notre arborescence de fichiers ressemble actuellement à cela :

```
# ll /mybtrfs1/
total 8
-rw-r--r-- 1 root root 18 mars  15 23:03 foo1
-rw-r--r-- 1 root root 24 mars  15 23:14 foo2
```

Soit **foo1** créé dans notre volume originel, puis **foo2** créé dans notre instantané. Remarquons la disparition du répertoire **/mybtrfs1/snapshot1** qui représentait l'existence de l'instantané **snapshot1** dans l'arborescence du volume originel.

Nous allons maintenant effectuer notre retour arrière très simplement, en démontant l'instantané actuellement en cours d'utilisation puis en remontant le volume originel :

```
# umount /mybtrfs1
# mount /dev/sda3 /mybtrfs1/
# ll /mybtrfs1/
total 8
-rw-r--r-- 1 root root 18 mars  15 23:03 foo1
dr-xr-xr-x 1 root root 34 mars  15 23:14 snapshot1
```

Notre arborescence est de retour à son état initial, en témoigne la disparition du fichier **foo2**. Si aucune option **subvol** ou **subvolid** n'est passée à la commande **mount**, c'est par défaut le volume dont l'identifiant est 0 qui est monté.

8.2 L'instantané devient l'original

À moins d'être terriblement malchanceux, vous n'aurez pas toujours besoin d'effectuer un retour arrière. Votre instantané est pour l'instant le sous-volume qui accueille les nouvelles données. La période de test étant passée, vous souhaiteriez que cet instantané devienne la nouvelle base pour votre arborescence de fichiers. Il est bien sûr très simple de répondre à cette demande, en indiquant notre instantané comme le nouveau volume de référence de la façon suivante :

```
# btrfs subvolume list /mybtrfs1
ID 268 top level 5 path snapshot1
root@btrfs:~# btrfs subvolume set-default 268 /mybtrfs1
```

Nous commençons par récupérer l'identifiant de notre sous-volume. Puis l'ordre **set-default** suivi par l'identifiant du sous-volume suivi du point de montage du volume original permet de déclarer un nouveau volume par défaut.

Qu'un instantané puisse devenir le nouveau sous-volume de référence offre une grande flexibilité sur nos manipulations de sous-volumes. En effet, l'ancien sous-volume de référence, après avoir cédé sa place à un nouveau venu, représente l'image de votre arborescence à un instant donné. Cette manipulation effectuée de façon régulière laissera sur votre système un historique - incarné par les différents sous-volumes - des états de votre arborescence, accessible en permanence et immédiatement réutilisable telle qu'elle était à ce moment-là, vous offrant la possibilité de naviguer parmi ces différents états, comme un développeur peut retrouver les différentes versions de son programme à l'aide d'un gestionnaire de sources.

9 Outils autour de btrfs

Mentionnons l'existence de deux outils de haut niveau pour gérer les instantanés.

Snapper **[2]**, développé par Arvin Schnell qui travaille pour SUSE. L'outil permet de gérer vos instantanés btrfs, de comparer leurs contenus, mais aussi d'effectuer des retours arrière.

Désormais intégrée à la distribution Fedora, la gestion des instantanés par le plugin Yum **yum-plugin-fs-snapshot [3]** vous permet de générer un instantané btrfs à chaque appel à la commande **Yum** avant chaque modification, vous permettant de simplement revenir en arrière si, par exemple, la mise à jour de votre système venait à échouer.

Les outils autour de btrfs ne sont pas encore légion mais devraient se multiplier avec sa popularisation, la flexibilité apportée par ce système de fichiers permettant d'envisager un système efficace de génération d'instantanés avant chaque modification potentiellement impactante pour vos systèmes en production.

Conclusion

Des fonctionnalités attendues depuis longtemps et une flexibilité permettant d'intégrer simplement btrfs à votre infrastructure existante sont les apports majeurs qu'attendaient les administrateurs système pour adopter btrfs. L'écart de fonctionnalités entre les systèmes de fichiers par défaut sur les distributions GNU/Linux ext3 ou ext4 et ceux en vogue sur d'autres systèmes comme ZFS, écart qui cristallisait auparavant les frustrations des utilisateurs, s'est réduit comme peau de chagrin. Simple à prendre en main, parfaitement intégré au système sur toutes les distributions majeures, btrfs gagne à être connu et utilisé. Gageons que la tendance de nombreuses solutions de stockage à dédaigner GNU/Linux pour profiter nativement de système de fichiers avec plus de fonctionnalités sur d'autres systèmes s'inversera bientôt. ■

Références

[1] http://kerneltrap.org/mailarchive/git-commits-head/2010/1/29/22722

[2] http://fr.opensuse.org/Portal:Snapper

[3] https://blogs.oracle.com/wim/entry/btrfs_root_and_yum_update

DÉVELOPPEZ AVEC JAVA SOUS OPENBSD

par Guillaume Dualé [OpenBSD addict !]

Amis développeurs du monde JAVA, vous rêvez d'un OS propre, stable, sécurisé, et qui rend jaloux tous vos collègues au boulot ? Ou plutôt vous êtes fatigué de cet Ubuntu avec Unity / GNOME Shell / TrucGraphiqueRelou ? Laissez-moi alors vous guider dans ce processus simple d'installation qui vous permettra d'avoir OpenBSD, GNOME 2, SUN JAVA et Netbeans. Un vrai desktop de dev' sur un OS super poilu :)

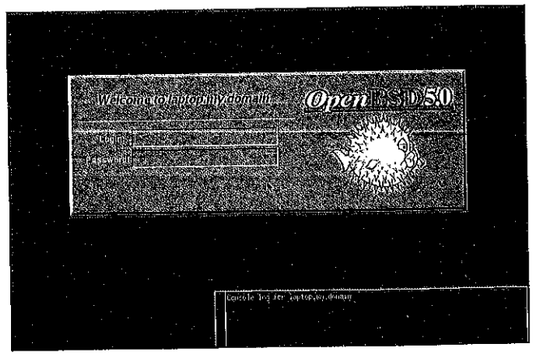
1 Introduction

La fiabilité et la sécurité d'OpenBSD n'étant plus à démontrer, je vous laisserai juge sur ce point. On me demande souvent pourquoi passer sous OpenBSD est-il bien. Pourquoi pas un autre BSD ou un autre système ? La réponse la plus simple est la suivante : le système d'exploitation qui répond à tous vos besoins est celui qu'il vous faut. Et dans le cas d'un environnement de développement amateur ou professionnel, il est à mon goût, un bon choix que de prendre OpenBSD comme *workstation*.

Dans un premier temps, il vous faudra OpenBSD avec Xorg installé sur votre machine.

L'installation d'OpenBSD n'est pas complexe et ce n'est pas le but de l'article de la décrire ici, je vous renvoie donc sur la documentation officielle française : http://www.openbsd.org/faq/fr/faq4.html.

Une fois que vous avez quelque chose comme ceci, vous pouvez continuer la lecture de cet article :



2 Installation de GNOME

La version de GNOME dans OpenBSD 5.0 est la 2.32.

Pour GNOME, il suffit de se servir du gestionnaire de *packages* **pkg_add** et d'installer les packages qui vont suivre.

Dans un premier temps, si ce n'est pas déjà fait, vous avez besoin d'exporter la variable **PKG_PATH** et de la faire pointer sur le serveur OpenBSD de votre choix.

En root, faites ceci par exemple :

```
export PKG_PATH=http://ftp.fr.openbsd.org/pub/OpenBSD/5.0/packages/i386/
```

N'oubliez pas le slash de fin de chaîne, et remplacez **i386** par votre architecture bien entendu. De plus, la variable **PKG_PATH** doit bien être en majuscule.

Une fois ceci fait, vous êtes fin prêt à installer GNOME :

```
pkg_add gnome-session gnome-backgrounds gnome-control-center gnome-panel gnome-themes-standard gnome-utils gnome-media gnome-menus gnome-screensaver gnome-system-monitor nautilus gdm
```

Vous pouvez ajouter ou supprimer des packages dans cette liste selon vos goûts.

Pour savoir quel package vous pourriez éventuellement ajouter, faites votre choix en faisant ceci :

```
pkg_info -Q gnome | grep -v installed
```

3 GDM au démarrage

Une fois que vous avez installé tout ça, si vous redémarrez, vous vous rendrez compte que rien n'a changé au niveau *login manager*, malgré le fait que nous ayons installé **gdm**.