Installer Ubuntu sur un système de fichiers F2FS

Table of Contents

- Installer Ubuntu sur un système de fichiers F2FS
  - Introduction
  - Comparaison des filesystem
    - Ext4 (quatrième système de fichiers étendu)
    - BtrFS (système de fichiers B-Tree)
    - XFS (système de fichiers étendu)
    - F2FS (système de fichiers compatible Flash)
- Installation de Ubuntu sur le système de fichiers F2FS
  - Prérequis
  - Créer la clé USB de boot
  - Créer une image esclave
  - Préparer le lecteur cible
    - Installer les outils
    - Partitionner le disque cible
  - Monter toutes les partitions
  - Copier les données de l'esclave dans la cible
  - Monter les système live sur la cible et chroot
  - Corriger le fichier fstab
  - Adapter initramfs et grub
  - Corrections pour la migration de la machine
    - Hostname
    - Réseau
  - Redémarrer

Introduction

F2FS (en anglais « flash-friendly file system », signifiant, « Système de fichier adapté au flash ») est un système de fichiers, créé par Kim Jaegeuk chez Samsung, destiné et orienté vers les mémoires flash NAND tels que les SSD ou bien les cartes eMMC, et cartes SD

Comparaison des filesystem

Lors du formatage de partitions sur un PC Linux, on dispose d'une grande variété d’options de système de fichiers. Cette section permet de comparer les systèmes de fichiers Btrfs, ext4, XFS, F2FS pour SSD.

Ext4 (quatrième système de fichiers étendu)

Ext4 est une version améliorée de l'ancien système de fichiers Ext3 qui comprend de nombreuses fonctionnalités intéressantes, notamment celles pour disques SSD (Solid State Drives).

La raison pour laquelle Ext4 est souvent recommandée est qu'il s'agit du système de fichiers le plus utilisé et le plus fiable sur Linux à ce jour. Il est utilisé dans les centres de données volumineux et en production, sur tous les types de disques durs, y compris les disques SSD. Il est recommandé lorsqu'on est peu soucieux des systèmes de fichiers.

Avantages	**Inconvénients**
Ext4 est aujourd'hui largement utilisé sur presque toutes les distributions Linux, et la plupart des utilisateurs de Linux sont familiarisés avec Ext4. Il n'est donc pas très difficile de trouver de l'aide lorsque vous l'utilisez sur votre disque SSD.	Ext4 étant basé sur une technologie plus ancienne, il lui manque les fonctionnalités de système de fichiers modernes présentes dans des systèmes tels que E2FS et BtrFS.
En plus de la prise en charge de TRIM, Ext4 inclut également de nombreuses autres optimisations SSD (pour les performances).	La journalisation sur Ext4 est activée par défaut, et les nouveaux utilisateurs ne sauront probablement pas comment la désactiver pour sauvegarder des lectures / écritures sur leurs disques SSD.
Les utilisateurs peuvent désactiver la journalisation pour protéger la nature limitée en lecture / écriture de leur disque SSD.	

BtrFS (système de fichiers B-Tree)

Btrfs est un système de fichiers assez solide pour une utilisation basique. Initialement conçu par Oracle Corporation pour être utilisé sous Linux, Btrfs est un nouveau type de système de fichiers créé pour Réparation simple.

Une des raisons pour lesquelles beaucoup de gens considèrent BtrFS pour un disque SSD est qu'il n'utilise pas de journal du système de fichiers, car sa journalisation ne lui permet pas d'économiser de l'espace en écriture (qui est limité sur les disques SSD). En outre, BtrFS dispose également d'une fonctionnalité d'instantané robuste, qui permet aux utilisateurs de créer (et d'annuler) des modifications sur le système instantanément.

Comme on pouvait s'y attendre, BtrFS prend en charge les fonctionnalités SSD habituelles telles que TRIM et d'autres optimisations SSD (telles que la défragmentation, etc.).

Avantages	**Inconvénients**
BtrFS n'a pas de journalisation activée par défaut. Par conséquent, contrairement à Ext4, vous n'aurez pas besoin de la désactiver si vous ne voulez pas que les journaux du système de fichiers réduisent votre taux de lecture/écriture.	BtrFS est extrêmement instable et risque de planter et de corrompre vos données en cas de problème.

Avantages	Inconvénients
Le système de fichiers étant nouveau et en cours de développement, de nouvelles fonctionnalités sont ajoutées régulièrement.	BtrFS a une fonctionnalité de copie sur écriture qui est sans doute aussi mauvaise que la journalisation du système de fichiers et qui pourrait (potentiellement) épuiser la limite de lecture / écriture de votre disque SSD.
BtrFS possède une fonctionnalité de défragmentation SSD qui permet aux utilisateurs de nettoyer les données sur leur disque.	

XFS (système de fichiers étendu)

Le système de fichiers XFS est reconnu pour sa capacité à traiter et gérer des éléments importants de la fiabilité, des performances et de la rapidité des données. De loin, XFS peut gérer des données volumineuses mieux que tout autre système de fichiers de cette liste et le faire de manière fiable. Donc, si vous avez beaucoup de données, devez y accéder rapidement et prévoyez de le stocker sur un SSD, XFS est un excellent choix. Lorsque vous installez un système d'exploitation Linux sur XFS sur un SSD, vous obtenez des fonctionnalités comparables à Ext4. tels que TRIM et d'autres optimisations.Vous bénéficierez également d'une fonction de défragmentation des disques SSD.

Avantages	Inconvénients
XFS est reconnu pour sa capacité à gérer facilement de grandes quantités de données.En utilisant XFS sur votre disque SSD, vous pouvez vous assurer que vos fichiers sont en sécurité.	XFS est un système de fichiers de journalisation, et il est impossible de désactiver cette fonctionnalité. Ne pas être en mesure de désactiver la journalisation est un sujet de méfiance si vous vous inquiétez de la limite de lecture/écriture de SSD.
Les avantages en termes de performances de XFS sur un disque SSD vous permettent de transférer et d'accéder aux fichiers et aux données beaucoup plus rapidement que d'autres systèmes de fichiers.	
XFS possède une fonctionnalité de défragmentation SSD très utile qui vous aidera à garder votre disque en bonne santé.	

F2FS (système de fichiers compatible Flash)

Le système de fichiers Flash-Friendly (F2FS) est un système de fichiers développé spécifiquement par Samsung pour les périphériques de stockage NAND sous Linux et les autres systèmes d'exploitation qui le prennent en charge. Cependant, de nombreux utilisateurs de Linux le craignent car toutes les distributions Linux ne le prennent pas en charge dans leur outil d'installation.

Avantages	Inconvénients
F2FS est explicitement conçu pour les disques SSD et autres périphériques de stockage flash, de sorte que votre système d'exploitation fonctionne efficacement et rapidement.	F2FS est un tout nouveau système de fichiers: certes, de nombreuses distributions Linux commencent à le prendre en charge, mais on ne peut pas dire que chaque système d'exploitation Linux facilite l'installation.

Avantages	Inconvénients
F2FS étant moderne et relativement nouveau, il disposera probablement de nouvelles fonctionnalités au fil du temps.	

Installation de Ubuntu sur le système de fichiers F2FS

L'installation d'Ubuntu directement sur F2FS depuis un ISO d'installation n'est pas possible car :

1. GRUB ne prend pas en charge F2FS et
2. Le programme d’installation Ubuntu ne peut pas s’installer sur F2FS.

Prérequis

Il faut avoir, dans aucun ordre particulier.

- Un disque esclave temporaire sur lequel installer initialement Ubuntu (`/dev/sdb` dans cet exemple)
- Un lecteur cible (`/dev/sda` dans cet exemple)
- Un disque/clé USB de démarrage sur lequel Ubuntu a été écrit pour démarrer. (`/Dev/sdc` dans cet exemple)

Créer la clé USB de boot

Si sdc est une clé USB,

```
dd if=ubuntu_xxxx.iso of=/dev/sdc bs=1M
```

Démarrer sur la clé USB et sélectionner le mode 'Try Ubuntu'

Créer une image esclave

Exécuter le programme d'installation graphique (Ubiquity)

Partitionner manuellement le disque esclave avec la partition ext2 de 250 Mo montée en tant que '/boot' et le reste avec ext4 monté '/', avec le chargeur de démarrage sous /dev/sdb.

Une fois installé, continuer d'essayer Ubuntu.

Préparer le lecteur cible

Installer les outils

```
sudo add-apt-repository universe
sudo apt-get update
sudo apt-get install f2fs-tools
```

Partitionner le disque cible

Démarrer GParted, ignorer l'erreur 'Le descripteur de pilote indique que la taille du bloc physique est de 2 048 octets, mais Linux indique 512'.

Créer une nouvelle table de partition 'msdos'

Créer deux partitions, d'abord 250 Mo de disque **ext2** (étiqueté dans l'exemple `Boot_ext2`), deuxième reste du disque **f2fs** (étiqueté dans mon exemple `Root_f2fs`).

Appliquer les modifications et définir les drapeaux 'flags' sur la partition ext2 sur 'boot'.

On peut ignorer les exclam rouges sur les f2fs nouvellement créés.

On doit maintenant avoir quelque chose comme ça, (les LABEL par défaut fournis par le programme d'installation seront quelque chose d'autre)

```
ubuntu@ubuntu:~$ sudo blkid
/dev/sdb1: LABEL="Boot" UUID="9fdca914-f565-4118-9417-c3a93fb111b0"
TYPE="ext2" PARTUUID="c0b4ef0c-01"
/dev/sdb2: LABEL="Root" UUID="b4dd9484-8f44-4d5e-9b69-2a9b1b3c5f08"
TYPE="ext4" PARTUUID="c0b4ef0c-02"
/dev/loop0: TYPE="squashfs"
/dev/sdc1: UUID="2015-04-22-12-30-17-00" LABEL="Ubuntu 15.04 amd64"
TYPE="iso9660" PTUUID="1cae5859" PTTYPE="dos" PARTUUID="1cae5859-01"
/dev/sdc2: SEC_TYPE="msdos" UUID="2474-67AF" TYPE="vfat"
PARTUUID="1cae5859-02"
/dev/sda1: LABEL="Boot_ext2" UUID="9033d4ad-37a6-4af4-95e0-5871033e7495"
TYPE="ext2" PARTUUID="40d2e08d-01"
/dev/sda2: LABEL="Root_f2fs" UUID="80422ab9-add2-454c-9930-7ee53288705a"
TYPE="f2fs" PARTUUID="40d2e08d-02"
```

Monter toutes les partitions

Dans Nautilus [Fichiers] le gestionnaire de fichier, cliquer sur les quatre partitions pour les monter

(Boot, Root, Boot*ext2, Boot*f2fs).

On doit avoir maintenant les partitions montées comme ça

```
/dev/sda2 sur /media/ubuntu/Root_f2fs type f2fs
(rw,nosuid,nodev,relatime,background_gc=on,user_xattr,acl,active_logs=6,uhel
per=udisks2)
/dev/sdb2 sur /media/ubuntu/Root type ext4
(rw,nosuid,nodev,relatime,data=ordered,uhelper=udisks2)
/dev/sdb1 sur /media/ubuntu/Boot type ext2
(rw,nosuid,nodev,relatime,uhelper=udisks2)
/dev/sda1 sur /media/ubuntu/Boot_ext2 type ext2
(rw,nosuid,nodev,relatime,uhelper=udisks2)
```

Copier les données de l'esclave dans la cible

Maintenant, il faut tout copier de l'esclave vers lacible, en faisant très attention de les prendre dans le bon sens! Cela ne prend que 4 minutes environ.

```
sudo rsync -avWHAX /media/ubuntu/Boot/ /media/ubuntu/Boot_ext2/
sudo rsync -avWHAX /media/ubuntu/Root/ /media/ubuntu/Root_f2fs/
```

Monter les système live sur la cible et chroot

Maintenant, faire mount bind des répertoires du système live dans la cible.

choisir les bons dossiers cibles montés sélectionner le lecteur cible correct `/dev/sd`X dans lequel le chargeur d'amorçage grub est en cours, dans ce cas, `/dev/sda`

```
sudo mount -o bind /dev /media/ubuntu/Root_f2fs/dev
sudo mount -o bind /sys /media/ubuntu/Root_f2fs/sys
sudo mount -o bind /proc /media/ubuntu/Root_f2fs/proc
sudo mount -o bind /media/ubuntu/Boot_ext2 /media/ubuntu/Root_f2fs/boot
sudo chroot /media/ubuntu/Root_f2fs
```

Corriger le fichier fstab

Maintenant il faut éditer le fstab cible

- Remplacer les deux UUID de l'esclave par les UUID de la cible
- Remplacer 'ext4' par 'f2fs' à la racine (/).
- Remplacer `errors=remount-ro` par `default` (attention, les options `f2fs` sont différentes des autres systèmes de fichiers)

Configuration d'origine

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sdc2 during installation
UUID=b4dd9484-8f44-4d5e-9b69-2a9b1b3c5f08 /               ext4
errors=remount-ro 0       1
# /boot was on /dev/sdc1 during installation
UUID=9fdca914-f565-4118-9417-c3a93fb111b0 /boot           ext2    defaults
0       2
```

Configuration de remplacement

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sdc2 during installation
UUID=80422ab9-add2-454c-9930-7ee53288705a /               f2fs    defaults 0
1
# /boot was on /dev/sdc1 during installation
UUID=9033d4ad-37a6-4af4-95e0-5871033e7495 /boot           ext2    defaults
0       2
```

Adapter initramfs et grub

Ajouter les modules `f2fs` à `initramfs` afin qu'il puisse voir la racine de `f2fs` lors du démarrage du système, puis installer grub sur le lecteur cible.

```
echo "f2fs" >>  /etc/initramfs-tools/modules
update-initramfs -u
grub-install /dev/sda
update-grub
```

Cela devrait produire le ci-dessous.

```
root@ubuntu:/# update-initramfs -u
update-initramfs: Generating /boot/initrd.img-3.19.0-15-generic
root@ubuntu:/# grub-install /dev/sda
Installing for i386-pc platform.
Installation finished. No error reported.
root@ubuntu:/# update-grub
Generating grub configuration file ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT
is set is no longer supported.
Found linux image: /boot/vmlinuz-3.19.0-15-generic
Found initrd image: /boot/initrd.img-3.19.0-15-generic
Found memtest86+ image: /memtest86+.elf
Found memtest86+ image: /memtest86+.bin
grub-probe: error: cannot find a GRUB drive for /dev/sdc1.  Check your
```

```
device.map.
Found Ubuntu 15.04 (15.04) on /dev/sdb2
done
root@ubuntu:/#
```

Ignorer l'erreur sur sdc car c'est la clé USB et cela créera également une entrée superflue sur le disque esclave (elle passera à toute mise à jour ultérieure du noyau grub)

Corrections pour la migration de la machine

Ceci s'applique uniquement si on veut déplacer ensuite le lecteur sur une autre machine ou si on crée plusieurs machines.

Hostname

Remplacer le nom d'hôte de la machine esclave par la cible dans `/etc/hosts` (par exemple, 127.0.1.1 TARGETX) et uniquement par le nom d'hôte dans `/etc/hostname`.

Réseau

Supprimer tout de `/etc/udev/rules.d/70-persistent-net.rules` (sinon, les périphériques réseau ne seront pas comme on le souhaite (eth0 sera eth1, etc.)

Redémarrer

Taper `Exit` deux fois, Éteindre, retirer la clé USB et le lecteur esclave et redémarrer dans le nouveau F2FS Ubuntu!

Si lors du redémarrage on obtient l'erreur **F2FS: Impossible de charger le pilote CRC32**

Il faudra ajouter un ou deux modules `crc32` à charger avec initramfs, crc32*pclmul et crc32*generic (avec un nouveau processeur Intel, `crc32_pclmul` devrait suffire, car il semble être seul le module en cours d'utilisation l'autre semble être un repli.)

```
/etc/initramfs-tools/modules
# List of modules that you want to include in your initramfs.
# They will be loaded at boot time in the order below.
#
# Syntax:  module_name [args ...]
#
# You must run update-initramfs(8) to effect this change.
#
# Examples:
#
```

```
# raid1
# sd_mod
f2fs
crc32_pclmul
crc32_generic
```

- Redémarrer sur la clef USB de boot
- Remonter le disque cible
- Remonter le système live sur le disque cible puis de nouveau chroot
- Ajouter dans `/etc/initramfs-tools/modules` les modules `crc32_plmul` et `crc32\genric` comme ci-dessus
- Reconstruire `initramfs` avec `sudo update-initramfs -u`
- Redémarrer.