Séances n° 6 et 7 : Stockage

1 Manipulation de systèmes de fichiers

Introduction

- Les partitions utilisables sous un système Linux peuvent ainsi provenir de plusieurs sources différentes, comme les partitions directement définies sur les disques durs, ou celles fournies par des systèmes tels que RAID ou LVM
- Quelle que soit la provenance d'une partition, l'utilisation est ensuite « transparente » sous Linux
- En particulier, il est nécessaire de choisir un système de gestion de fichiers (ext3, ext4, btrfs, reiserfs, jfs, xfs...), de formater la partition, de la monter pour pouvoir l'utiliser....

Création de système de gestion de fichiers (SGF)

- Linux sait gérer de nombreux systèmes de gestion de fichiers, même si les plus usités sont généralement ext3 et ext4.
- Sous Linux, la commande `mkfs` est la commande générique permettant de créer un SGF, dont le type est précisé au moyen du paramètre `-t`
- Cette commande sous-traite alors la requête à une autre commande de la forme `mkfs.xxxx`, où `xxxx` est le type du SGF
- Exemple : `mkfs -t jfs /dev/sda1` ou `mkfs.jfs /dev/sda1`
- Les commandes de création de SGF acceptent souvent de nombreux paramètres techniques, permettant de préciser finement certaines caractéristiques, intéressantes lorsque l'usage de la partition concernée est « atypique »

Montage des SGF

- Pour pouvoir être utilisés sous Linux, les partitions doivent être préalablement *montées*
- Le *montage* est l'opération qui consiste à associer un répertoire (appelé *point de montage*) à une partition
- Une fois qu'une partition est montée, les données qu'elle contient sont accessibles à partir du point du montage
- L'opération de montage fait abstraction du support des SGF, qui peuvent aussi bien se trouver sur un disque local, que sur le réseau ou en mémoire...
- La commande permettant de réaliser des montages est la commande `mount`, dont la syntaxe générale est `mount [-t type] périphérique répertoire`
- L'option `-t` permet de préciser le type de SGF contenu sur la partition à monter, type généralement déterminé automatiquement par Linux. Pour information, les types les plus usités sont :
  - `ext2`, pour les systèmes de fichiers ext2 ;
  - `ext3`, pour les systèmes de fichiers ext3 ;
  - `ext4`, pour les systèmes de fichiers ext4 ;

  - `reiserfs`, pour les systèmes de fichiers ReiserFS ;
  - `iso9660`, pour les CD-ROM (qu'ils soient avec extensions Joliet ou Rock Ridge ou en mode ISO 9660 pur) ;
  - `ntfs`, pour les systèmes de fichiers NTFS ;
  - `smbfs` et `cifs`, pour les systèmes de fichiers distribués par Samba ;
  - `msdos`, pour les systèmes de fichiers FAT normaux ;
  - `vfat`, pour les systèmes de fichiers FAT32.
- Si le répertoire de montage n'est initialement pas vide, les fichiers qui s'y trouvent sont masqués par le SGF monté

Démontage des SGF

- L'opération antagoniste du montage est le *démontage*
- Il est indispensable d'effectuer cette opération lorsqu'une partition n'est plus utilisée : le démontage permet d'alerter le système d'exploitation, de vider les *buffers* en cours d'utilisation avec cette partition et d'effectuer toute synchronisation nécessaire
- Le non démontage d'une partition peut provoquer des pertes de données irrémédiables sur cette partition...
- La commande permettant de démonter une partition est la commande `umount`, dont la syntaxe est `umount périphérique` ou `umount répertoire`

Configuration du montage et du démontage des SGF

- Les opérations de montage et de démontage de partitions peuvent vite se révéler assez fastidieuses
- Il est de plus nécessaire de se doter de mécanismes d'automatisation, notamment lors du démarrage ou de l'arrêt d'une machine...
- Le fichier `/etc/fstab` permet de stocker les configurations relatives à chaque partition (périphérique, point de montage, type de SGF entre autres)
- Il permet le montage et le démontage automatique de certaines partitions lors du démarrage et de l'arrêt des machines, ainsi que des utilisations simplifiées de la commande `mount` lorsque la partition concernée est répertoriée
- Le fichier `/etc/fstab` contient une ligne pour chaque partition répertoriée comportant 6 colonnes :
  - le fichier spécial permettant d'accéder au système de fichiers ;
  - le répertoire servant de point de montage par défaut ;
  - le type du système de fichiers ;
  - les options de montage pour ce système de fichiers ;
  - un entier indiquant si le système de fichiers doit être sauvegardé ;
  - un entier indiquant l'ordre que ce système de fichiers doit avoir dans la liste des systèmes de fichiers à vérifier.
- Grâce à ces informations, la syntaxe de la commande `mount` est alors plus simple : `mount périphérique` ou `mount répertoire`
- Les principales options possibles pour le montage sont :
  - l'option `defaults`, qui permet de choisir les options par défaut pour ce système de fichiers (c'est-à-dire `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, et `async`) ;
  - l'option `auto`, qui permet de faire en sorte que le système de fichiers soit monté automatiquement au démarrage du système ;

— l'option `user`, qui permet d'autoriser le montage de ce système de fichiers par les utilisateurs (et donc pas seulement par l'administrateur du système) ;
— l'option `ro`, qui permet de monter le système de fichiers en lecture seule ;
— l'option `rw`, qui permet de monter le système de fichiers en lecture et écriture ;
— l'option `exec`, qui permet d'autoriser l'exécution des fichiers exécutables sur ce système de fichiers si celui-ci ne supporte pas la notion de droit d'exécution ;
— l'option `uid=utilisateur`, qui permet de spécifier le numéro utilisateur de l'utilisateur propriétaire du répertoire racine de ce système de fichiers ;
— l'option `gid=groupe`, qui permet de spécifier le numéro groupe du groupe d'utilisateurs auquel le répertoire racine du système de fichiers appartient ;
— l'option `mode=valeur`, qui permet de fixer les droits sur le répertoire racine du système de fichiers à monter. La valeur valeur est donnée en octal ;
— l'option `umask=valeur`, qui permet de fixer les droits sur les fichiers qui ne sont pas gérés par le système de fichiers. La valeur valeur est donnée en octal ;
— les options `codepage=cp` et `iocharset=charset`, qui permettent de fixer les tables de conversion des caractères pour les systèmes de fichiers pour lesquels les noms de fichiers doivent être transcodés.

Vérification des SGF

— Cette opération ne devrait jamais être effectuée, Linux prenant lui-même l'initiative de vérifier systématiquement les SGF posant des problèmes (non démontés avant un arrêt de la machine) et les vérifiant périodiquement sinon (tous les n démarrage par exemple)
— Cependant, certaines situations marginales peuvent nécessiter une vérification manuelle d'une partition
— Un SGF ne se manipule en général que lorsqu'il est démonté. Si le SGF concerné est celui hébergeant la partition racine `/`, il faut démarrer la machine depuis un support amovible (clé USB, CDROM) contenant des outils de vérification, ou monter ce SGF en lecture seule
— La commande permettant de vérifier un SGF est la commande `fsck` qui sous-traite l'opération, comme dans le cas de création de SGF, à une commande de la forme `fsck.xxxx` où `xxxx` est le type du SGF
— La syntaxe standard d'utilisation de cette commande est `fsck -a périphérique`, comme par exemple `fsck -a /dev/sda5`
— Une fois la vérification terminée, et les erreurs corrigées (!), la partition correspondante peut alors être remontée sous le système
— La commande `dumpe2fs` permet de consulter différentes informations sur un système de fichiers ext2/3/4.

2 RAID

Si l'utilisation d'un seul disque dur dans une machine suffit dans de nombreux cas (PC portable, ordinateur de bureau), il est souvent nécessaire de combiner plusieurs disques afin d'augmenter la fiabilité ou les performances de l'installation.

La combinaison de différents disques durs se fait généralement avec le système RAID (*Redundant Array of Inexpensive Disks*). Il existe différentes manières de combiner les disques

durs, et donc différents niveaux de RAID :

- RAID 0 : permet de répartir une information sur plusieurs disques à la fois et ainsi d'accélérer très sensiblement les requêtes d'entrées comme de sorties. La perte d'un disque induit généralement la perte de toutes les données.
- RAID 1 : permet de dupliquer les informations stockées sur 2 disques, les deux disques étant mutuellement *miroir* de l'autre. Le gain est nul sur les écritures, mais est appréciable sur les lectures (les deux disques sont mis à contribution). La perte d'un des disques n'entraîne aucun problème de disponibilité : l'autre disque assure seul le relais, le temps que l'autre disque soit remplacé.
- RAID 5 : les données sont réparties sur plusieurs disques comme en RAID 0, mais on stocke aussi des informations de parité (typiquement, un *Xor* entre les données) permettant de reconstruire les données dans le cas où un disque deviendrait indisponible. Ces informations de parité sont stockées alternativement sur les différents disques afin de maximiser les performances.
- RAID 6 : même principe que le niveau 5, mais les informations de parité sont présentes 2 fois (ou plus). Cela permet de résister à la défaillance de plus d'un disque.

Le RAID peut être implémenté soit au niveau matériel (dans la carte contrôleur de disques durs), soit au niveau logiciel (dans le noyau Linux). L'utilisation de RAID matériel permet de meilleures performances en évitant de charger le processeur principal avec les calculs de parité (en *Raid* 5 ou 6). L'utilisation de RAID logiciel est plus facile et plus souple.

Dans le cas du RAID matériel, l'ensemble des disques apparait comme un seul disque de grande capacité, et la configuration se fait avec des outils dépendant du constructeur de la carte RAID :

```
SCSI device sdc: 27334230016 512-byte hdwr sectors (13995126 MB)
```

Sous Linux, la configuration du RAID logiciel se fait avec `mdadm` (MD = Multi Device). `/proc/mdstat` permet de consulter le statut actuel des différents disques RAID.

Exemple 1 :

```
md1 : active raid1 sdb2[1] sda2[0]
      136448 blocks [2/2] [UU]

md2 : active raid1 sdb3[1] sda3[0]
      129596288 blocks [2/2] [UU]

md3 : active raid5 sdl1[9] sdk1[8] sdj1[7] sdi1[6] sdh1[5] sdg1[4] sdf1[3]
                                                    sde1[2] sdd1[1] sdc1[0]
      1318680576 blocks level 5, 1024k chunk, algorithm 2 [10/10] [UUUUUUUUUU]

md0 : active raid1 sdb1[1] sda1[0]
      16787776 blocks [2/2] [UU]
```

Exemple 2 :

```
Personalities : [raid1] [raid6] [raid5] [raid4]
md127 : active raid5 sdh1[6] sdg1[4] sdf1[3] sde1[2] sdd1[1] sdc1[0]
      1464725760 blocks level 5, 64k chunk, algorithm 2 [6/5] [UUUUU_]
```

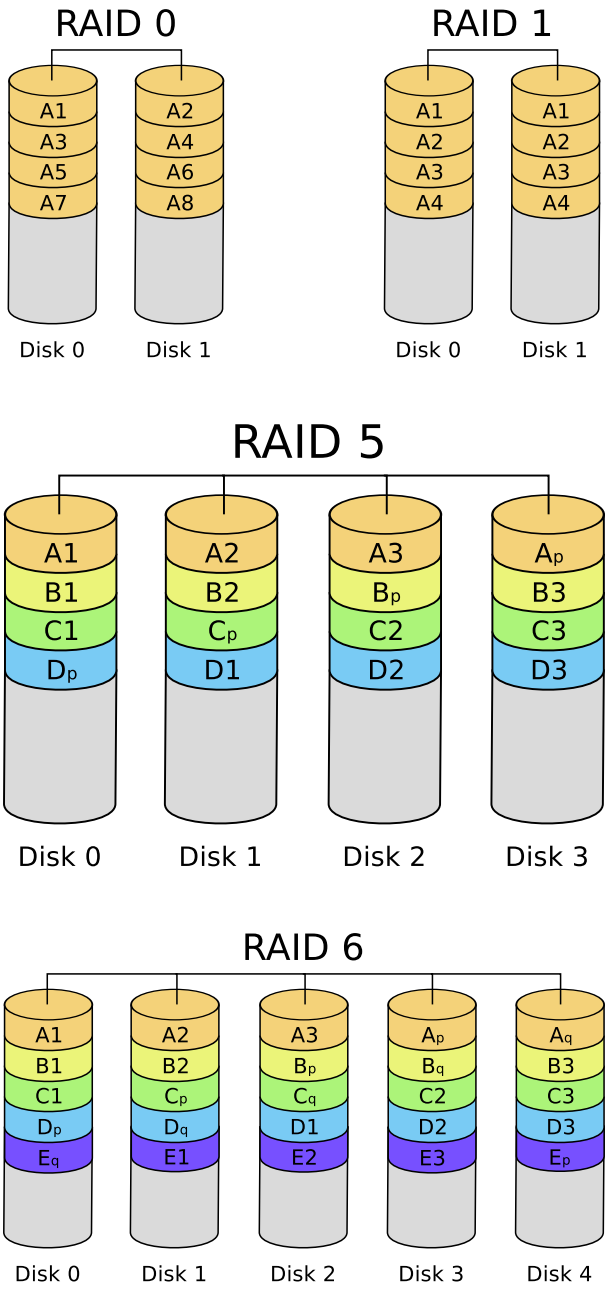

FIGURE 1 – Différents niveaux de RAID (images Wikipédia)

```
[=>.....]  recovery = 12.6% (37043392/292945152) finish=127.5min
                                                  speed=33440K/sec
```

Q1. Quels niveaux de RAID faut-il utiliser dans les cas suivants ?

(a) Disque "système" d'un serveur : espace de stockage de taille limitée (moins de 100 Go), mais il faut absolument éviter de perdre des données. On cherche une solution simple, puisque l'espace de stockage est de taille limitée.

(b) Espace de stockage temporaire "scratch" : gros espace (10 To) où les utilisateurs font stocker des données temporaires. La seule priorité est d'être le plus performant possible. La perte de données est ennuyeuse, mais les utilisateurs sont prévenus, et préfèrent privilégier les performances.

(c) Espace de stockage "homes" : gros espace (10 To) où les utilisateurs vont stocker leurs données personnelles. Il faut un bon compromis entre performances (cet espace de stockage va être fréquemment utilisé) et fiabilité.

Q2. Installez les outils permettant de configurer du RAID logiciel.

Q3. Exécutez les commandes suivantes (en tant que root) pour créer 4 block devices de 100 Mo. Nous allons utiliser ces *block devices* pour simuler des disques durs dans la suite du TP.

```
modprobe loop
for i in $(seq 1 4); do
dd if=/dev/zero of=/tmp/blockdev$i bs=1M count=100
losetup /dev/loop$i /tmp/blockdev$i
done
losetup -a # liste les block devices créés
```

Q4. Pendant les questions suivantes, il est utile de surveiller le contenu de `/proc/mdstat`. Ouvrez un autre terminal, et lancez `watch -n 1 cat /proc/mdstat`. Il est aussi utile de lancer `dmesg` pour regarder les messages générés par le noyau lors de la création des différentes solutions de RAID.

Q5. Utilisez les block devices `/dev/loop1` à `/dev/loop4` pour configurer différents types de RAID logiciel. À chaque fois, créez un système de fichier `ext4` sur le volume RAID créé, montez-le, et regardez l'espace disque disponible. Après chaque étape, il est nécessaire de démonter le système de fichier et d'arrêter le volume RAID pour pouvoir passer à l'étape suivante.

(a) RAID 0 sur 2 disques.

(b) RAID 1 sur 2 disques. Le créer d'abord sur un disque (en utilisant `--force` si nécessaire), puis créer et monter le système de fichiers, puis rajouter le deuxième disque. Attention, dans ce cas, le disque est d'abord ajouté au RAID en *spare*. Il faut utiliser `--grow --raid-devices=2` pour étendre le RAID 1 sur les deux disques.

(c) RAID 5 sur 4 disques.

(d) RAID 6 sur 4 disques.

Q6. Sur votre RAID 6, marquez un des disques comme *faulty* avec `mdadm /dev/md0 --fail /dev/loop1`. Essayez de faire des manipulations sur votre système de fichier (créez un

fichier), puis utilisez la commande `sync` pour forcer l'écriture des modifications sur le disque, et consultez les messages d'erreur du noyau (`dmesg | tail`) à la recherche d'erreurs concernant votre stockage. Que devriez-vous constater ? Que constatez-vous ?

Q7. Sur votre RAID 6, marquez un deuxième disque comme *faulty*. Essayez de faire des manipulations sur votre système de fichier (en utilisant `sync`). Que constatez-vous ?

Q8. Sur votre RAID 6, marquez un troisième disque comme *faulty*. Essayez de faire des manipulations sur votre système de fichier (en utilisant `sync`). Que constatez-vous ?

Question subsidiaire :

Q9. Quels sont les changements possibles sur un RAID ? Par exemple, est-il possible de passer d'un RAID 1 sur 2 disques à un RAID 5 sur 3 disques ? et ensuite à un RAID 6 sur 4 disques ? Vérifiez.

3 Gestion fine des espaces de stockage : LVM

La gestion des espaces de stockage à base de disques durs et de partitions est très peu souple. Par exemple, il est difficile de redimensionner une partition, puisque cela nécessite forcément de revoir les dimensions de l'ensemble des partitions. LVM (*Logical Volume Manager*) est une solution à ce problème.

LVM reprend certaines des fonctionnalités du RAID (logiciel !) et en propose aussi de nouvelles. Il peut être utilisé seul ou au dessus d'une solution RAID, matérielle ou logicielle.

LVM propose une gestion du stockage de données organisée autour de 3 concepts différents

— les *physical volumes* (PV), couche la plus « basse », correspondant à un disque dur ou à ses partitions, ou d'une manière générale ou tout ce qui y ressemble (y compris un disque RAID),

— les *logical volumes* (LV), couche la plus « haute », correspondant aux partitions finalement accessibles depuis le système. Ces volumes logiques hébergent donc les systèmes de gestion de fichiers et ont la particularité, puisqu'ils sont gérés par LVM, d'être redimensionnables (réduction ou extension).

— les *volume groups* (VG) rassemblent les PV et LV dans un même groupe. Ainsi, Un VG va contenir un ensemble de PV qui seront utilisés pour créer un ensemble de LV. Mais les frontières entre PV ne correspondent généralement pas forcément aux frontières des LV.

Commandes liées à LVM

Générales

— `lvmdiskscan` : liste toutes les partitions accessibles et précise lesquelles sont utilisations par LVM

Relatives aux volumes physiques

- `pvscan` : liste tous les volumes physiques disponibles sur le système
- `pvcreate` : initialisation d'une partition afin de pouvoir l'utiliser comme un volume physique
- `pvdisplay` : affiche des information liée à un volume physique
- `pvmove` : déplace les données d'un volume physique dans un autre volume physique appartenant au même volume groupe

Relatives aux groupes de volumes

- `vgscan` : liste tous les groupes de volumes disponibles sur le système
- `vgcreate` : création d'un volume groupe à partir de 1 ou plusieurs volumes physiques. Le nouveau volume groupe sera accessible sous `/dev/`*nomVG*
- `vgextend` : ajout d'un volume physique dans un volume groupe
- `vgreduce` : suppression d'un volume physique dans un volume groupe
- `vgdisplay` : affiche des informations liées à un volume groupe
- `vgchange` : active ou désactive un volume groupe
- `vgremove` : supprime un volume logique, devant préalablement désactivé et vidé de tout volume logique

Relatives aux volumes logiques

- `lvscan` : liste tous les volumes logiques disponibles sur le système
- `lvcreate` : création d'un volume logique dans un volume groupe. Le nouveau volume logique sera accessible sous `/dev/`*nomVG*`/`*nomLV*
- `lvextend` : extension de la taille d'un volume logique
- `lvreduce` : réduction de la taille d'un volume logique
- `lvdisplay` : affiche des informations liées à un volume logique
- `lvrename` : renomme un volume logique
- `lvremove` : supprime un volume logique

Questions

Q1. Installez les logiciels permettant de gérer les volumes LVM.

Q2. Lors de l'installation de votre machine, vous avez créé deux partitions annexes, actuellement vides. Transformez ces deux partitions en volumes physiques LVM (commande `pvcreate`).

Q3. Créez un groupe de volumes LVM à partir de ces deux partitions (commande `vgcreate`)

Q4. Consultez la liste des PV et des VG (commandes `pvdisplay` et `vgdisplay`). De combien d'espace disponible disposez-vous dans votre VG ?

Q5. Créez 2 volumes logiques dans votre VG, en laissant de l'espace disque disponible dans le VG (commande `lvcreate`).

Q6. Listez les LV (commande `lvdisplay`).

Q7. Créez un système de fichier `ext4` sur chacun de ces 2 LV. Montez les.

Q8. Agrandissez un des LV. Le système de fichier est-il redimensionné ? Sinon, redimensionnez-le (avec `resize2fs`).

Q9. Réduisez la taille du LV et du système de fichier. Est-ce possible sans démonter le système de fichier ?

Q10. Nettoyez votre système. Supprimez l'ensemble des LV, VG, PV créés.

4 Visualisation des fichiers ouverts

Il est souvent utile de savoir quels sont les fichiers actuellement ouverts, par exemple pour tuer les processus les utilisant avant de démonter une partition. Les commandes `lsof` et `fuser` permettent de réaliser cette opération.

Q1. Avec `fuser`, obtenez un listing détaillé (nom du processus, utilisateur) des processus utilisant des fichiers dans `/home`.

Q2. Avec `lsof`, listez tous les fichiers ouverts sur le système.

Q3. Avec `lsof`, listez toutes les connexions réseaux ouvertes sur le système.

5 Questions subsidiaires

Q1. (Inspiré d'un cas réel) Sur l'une des deux partitions annexes (qui fera office de disque dur), créez un PV, un VG et un LV, ainsi qu'un système de fichiers. Montez-le. Imaginez maintenant que vous rendez compte que la machine concernée a en fait deux disques (dans notre cas, la deuxième partition annexe), et qu'il faudrait rajouter du RAID 1 en-dessous du LVM ! Comment procéderiez-vous ? Indice : il existe une solution permettant de réaliser le changement sans avoir à recopier les données présentes sur la partition ailleurs, et même sans démonter le système de fichiers. Réalisez la manipulation.

Q2. Un nouveau système de fichiers pour Linux est en cours de développement : BTRFS. Quels sont ses avantages ? Configurez un volume BTRFS sur votre machine.

Q3. Le système de fichier ZFS, originellement développé par Oracle sur Solaris, est perçu comme révolutionnaire. Essayez le, et comparez le aux solutions Linux de RAID logiciel et de LVM. Vous pouvez :

(a) Installer FreeBSD (ou Debian GNU/kFreeBSD), puisque ZFS est intégré au noyau FreeBSD ;

(b) Utiliser le portage Linux de ZFS du projet *ZFS on Linux* (http://zfsonlinux.org/)