Gestion de disques

Michel Blockelet

5 Janvier 2010[1][2]

[1] Bonne année !

[2] Version corrigée par rapport à quelques remarques effectuées pendant le séminaire.

Vocabulaire

Un disque

- Disque = MBR + n partitions primaires, $n \leq 4$
- MBR (Master Boot Record) : contient la table de partitions et du code de boot
- MBR : par défaut, un code qui cherche la première partition marquée “amorçable” ; mais on peut y mettre GRUB par exemple
- Partition : découpage d’un disque
- Partition étendue : permet de stocker n partitions logiques dans 1 partition primaire (n théoriquement illimité, mais pouvant être limité selon les OS)

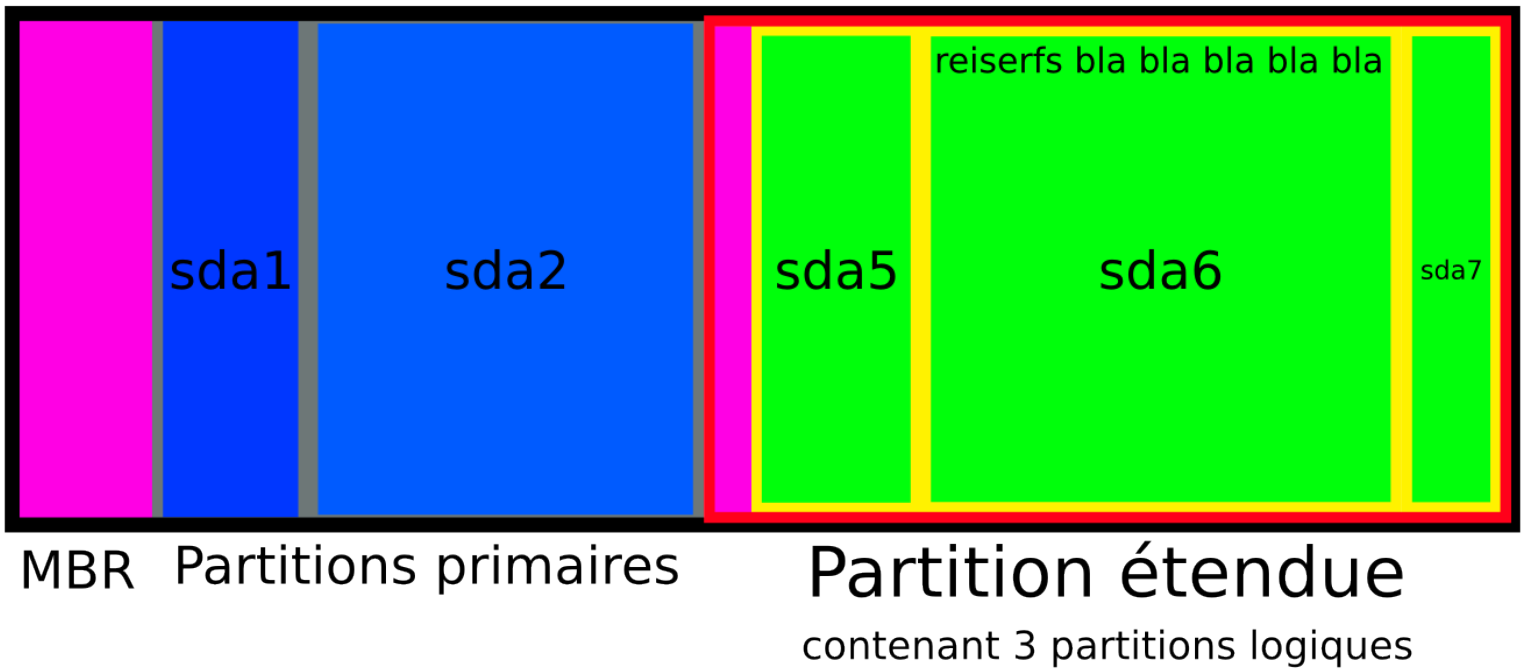
Vocabulaire

Systèmes de fichiers

- Système de fichiers : système contenu dans une partition (ou un block device), permettant de stocker et retrouver des fichiers
- Caractéristiques différentes : vitesse, taille des fichiers, gestion des permissions, journalisation, compression, sauvegardes
- Peut être contenu dans un fichier (exemples : `.iso` contenant un système de fichiers iso9660, squashfs système fixe compressé (livecds) ; mais on peut aussi avoir de l'ext3 ou n'importe quoi d'autre)
- Possède en général un UUID (identificateur moche généré aléatoirement) et peut avoir un Label (étiquette, nom donné par l'utilisateur)

Le schéma qui va bien

Mon disque :



Pourquoi ?

Problématiques

- Gestion ardue : par exemple, pour fusionner deux espaces non partitionnés avant et après une partition, il faut déplacer les données au milieu
- Système de fichiers limité à l'espace du disque dur
- Peu d'"automatisation" quant à la création de nouvelles partitions

Buts

- Créer, gérer facilement des "disques virtuels"
- Aggréger des disques durs
- Sécuriser les données
- Faciliter la création de domU

Block devices

Sous Linux

- Block device : représentation sous forme de "fichier" d'un disque
- `/dev/sda` représente le disque, `/dev/sda1` une partition
- Par exemple, `/dev/sdX` pour des disques SCSI (ou SATA), `/dev/hdX` pour des disques IDE, `/dev/mmcblkX` pour des cartes mémoire, ...
- `/dev/mapper/` contient les disques "spéciaux" (créés par dm-crypt, LVM)
- `/dev/disk/` contient des liens symboliques vers les disques (`by-id`, `by-label`, `by-path`, `by-uuid`)

Utilitaires sous Linux

Utilitaires

- `fdisk` : permet de créer, modifier, lister les partitions d'un block device
- (g)`parted` : permet de modifier les partitions et les systèmes de fichiers contenus (frontend aux différents utilitaires des différents systèmes de fichiers)
- `mkfs.*` : liens vers les divers utilitaires de création d'un système de fichiers
- `fsck.*` : liens vers les divers utilitaires de vérification d'un système de fichiers (utilisés au démarrage)
- `dd` : copier (une partie d')un disque

Utilitaires sous Linux

Gestion d'un système Ext3

- `mke2fs`, `e2fsck` : création / vérification d'un système ext2/3/4
- `resize2fs` : redimensionnement d'un système ext2/3/4 ; par défaut, agrandit vers tout l'espace disponible
- `tune2fs` : modification des caractéristiques d'un système ext2/3/4 ; par exemple, ajout d'un journal, modification de l'intervalle de vérification, ...

Principe de LVM

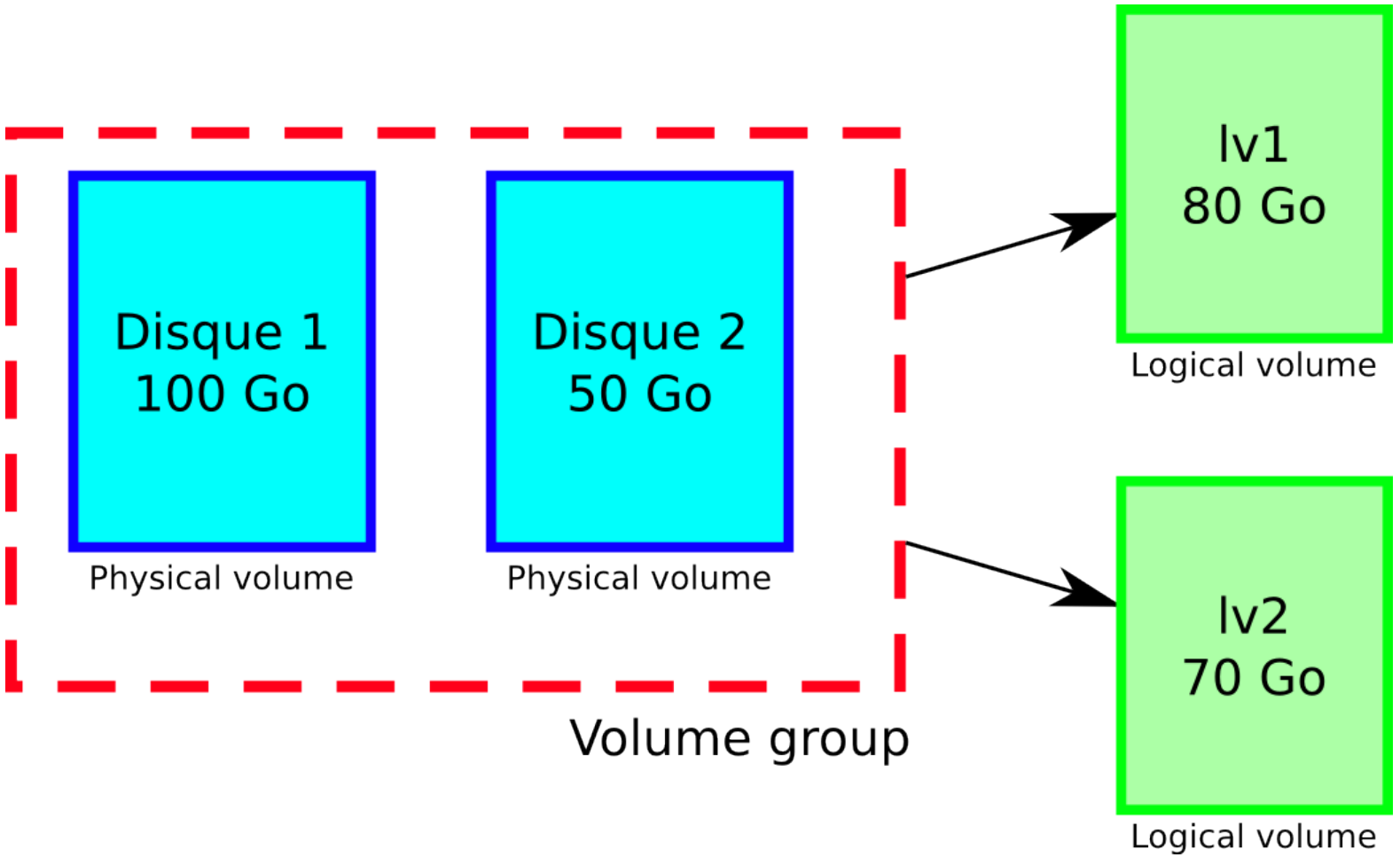
LVM

- LVM : Logical Volume Manager
- Permet d'aggréger des disques et d'y créer des volumes logiques
- Gère automatiquement l'emplacement des données de ces volumes

Fonctionnement

- Physical Volumes : volumes qu'on aggrège (typiquement, les disques durs)
- Volume Group : résultat de l'aggrégation de Physical Volumes, un gros bloc dans lequel on peut créer des Logical Volumes
- Logical Volumes : bloc d'espace que l'on peut ensuite utiliser à notre convenance

Schéma


Disque 1
100 Go
Disque 2
50 Go
Physical volume
Physical volume
Volume group
lv1
80 Go
Logical volume
lv2
70 Go
Logical volume

Avantages

Avantages

- Fusion de plusieurs disques
- Nombre de volumes (quasi-)illimité
- Gestion simplifiée : “je veux un volume de 20 Go” et LVM s’occupe de trouver l’espace pour le créer
- Contrairement au RAID 0, la défaillance d’un disque n’entraîne que la perte des données qui étaient situées sur ce disque (les secteurs des volumes situés sur ce disques seront vus comme des secteurs défectueux)

“Inconvénients”

- Couche supplémentaire dans l’accès aux données (mais négligeable)
- Ne marche pas sous tous les systèmes d’exploitation !

Commandes

Commandes LVM

- `pvs`, `vgs`, `lvs` : Lister les Physical Volumes / les Volume Groups / les Logical Volumes
- `pvcreate monpv` : Initialiser un disque en tant que Physical Volume pour utilisation avec LVM
- `vgcreate supervg` : Créer un Volume Group
- `vgextend supervg monpv` : Ajouter un Physical Volume à un Volume Group
- `lvcreate -L 20G -n level1 supervg` : Créer un Logical Volume nommé `level1`
- `lvextend -L +10G supervg/level1` : Ajouter de l'espace à un Logical Volume

Commandes

Et après ...

- `mkfs.ext3 /dev/supervg/level1` : Créer un système ext3 sur le LV level1
- `e2resize /dev/supervg/level1` : Agrandir le système pour utiliser tout l'espace disponible

Principe du RAID

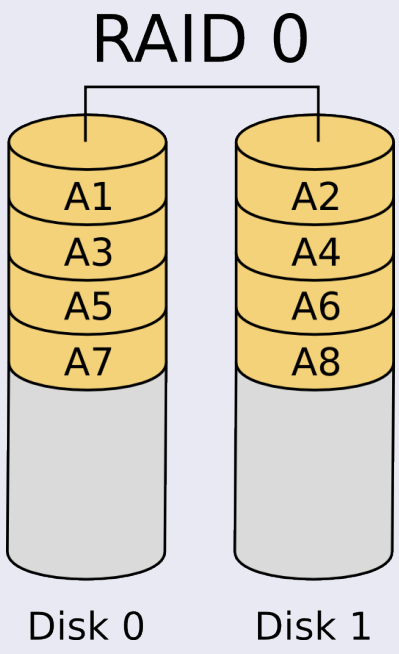
RAID

- RAID : Redundant Array of Inexpensive Disks
- "Aggrégation avec redondance de disques peu coûteux" (et donc peu fiables)
- Permet d'aggréger plusieurs disques de même taille de façon à garantir la sécurité des données en cas de défaillance d'un disque (sauf le RAID 0)

Niveaux de RAID

RAID 0

RAID 0 (ou Stripe) : Répartit les données entre n disques

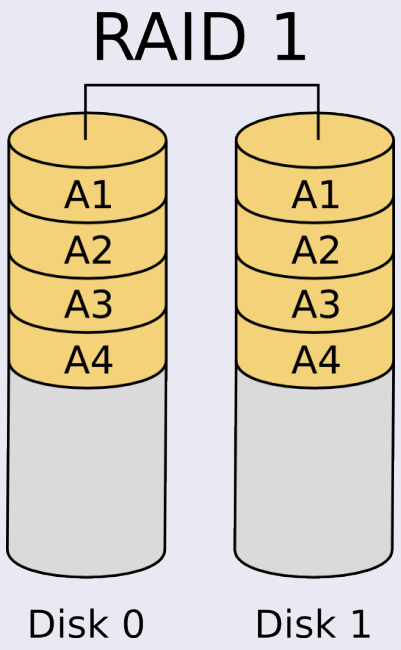


Données : $a_1 b_1 a_2 b_2$
Avantage : Lecture et écriture n fois plus rapides (il lit en parallèle sur les n disques)
Inconvénient : Si un disque lâche, tout est perdu ...

Niveaux de RAID

RAID 1

RAID 1 (ou Mirror) : Copie les données de façon identique entre les n disques

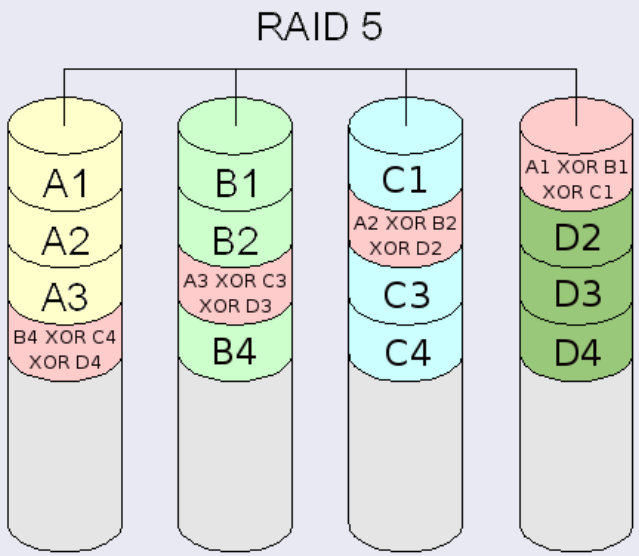


Avantages : Lecture n fois plus rapide, sécurité en cas de défaillance de $n - 1$ disques

Niveaux de RAID

RAID 5

RAID 5 : RAID 0 avec un bit de parité supplémentaire partagé entre les disques



Données : $a_1 b_1 c_1 a_2 b_2 d_2 \ldots$
Caractéristique : $d_i = a_i \oplus b_i \oplus c_i$ (et donc $a_i = b_i \oplus c_i \oplus d_i$)
Avantages : vitesse, espace, sécurité

Utilisation d'un RAID

RAID Hardware

- Décharge les calculs sur une puce plutôt que sur le CPU
- Rarement contrôlable depuis l'OS
- Format spécifique au contrôleur ; ça ne marche pas en changeant le contrôleur

Mdadm

- Mdadm : outil de gestion de RAIDs software sous Linux
- Fonctionne à partir des block devices, peut donc fonctionner à partir de n'importe quoi (SATA, IDE, iSCSI, ...)
- Calculs effectués par le processeur, mais impact pas si important

Utilisation de Mdadm

Tout d'abord ...

- `man mdadm` est votre ami, vraiment !
- *Votre cerveau est votre ami aussi !*

Principes généraux

- Crée des arrays `/dev/mdX`
- Chaque composant de l'array est composé d'un superblock et ensuite de la zone de données
- Le superblock permet de savoir à quel array ça appartient, et quel est le numéro d'ordre du disque
- On peut reconstituer (automatiquement) un array en lisant juste les superblocks

Utilisation de Mdadm

Faulty, spare

- **Faulty** : disque défaillant
- **Spare** : disque supplémentaire à un RAID (par exemple un 5ème disque dans un RAID de 4 disques)
- Quand un disque présente des erreurs (impossible d'accéder à un secteur, disque disparu, ...), il est marqué comme faulty et retiré du RAID
- Si le RAID a un disque spare, il est utilisé pour remplacer le disque faulty, et resynchronisé
- Si le RAID n'a pas disque spare mais a assez de tolérance, il continue à fonctionner

mdstat

mdstat

/proc/mdstat : Fichier contenant le statut du RAID

```
md1 : active raid5 sdh[0] sdd[5] sdc[4] sda[3] sdb[2] sdg[1]
          4883812480 blocks level 5, 64k chunk, algorithm 2 [6/6] [UUUUUU]
          [===>.................] resync = 18.5% (180987520/976762496) finish=1850.3min speed=7166K/sec
md0 : active raid1 hdd(F) hdd[1]
          156290816 blocks [2/2] [_U]
```

Indique les RAIDs actifs, les disques attachés, les disques manquants, leur état de resynchronisation, ...

Utilisation de mdadm

Modes de fonctionnement

- Différents “modes de fonctionnement” (ou opérations) :
- **Assemble** : Assembler les composants d'un RAID déjà existant
- **Build** : Créer/assembler un RAID sans superblocks
- **Create** : Créer un RAID avec superblocks
- **Monitor** : Surveiller l'activité du RAID (DegradedEvents ...)
- **Grow** : Modifier un RAID existant
- **Incremental** : Ajouter les disques un par un à un RAID

Créer un array

Créer un array

- `mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/hda /dev/hdb`
- On crée un array RAID 1 /dev/md0, avec 2 disques `/dev/hda` et `/dev/hdb`

Créer un array

Situation

Données sur un troisième disque, seulement deux disques de libre, on veut créer un RAID 5 avec les trois disques et les données

Créer un array non complet

- `mdadm --create /dev/md1 --level=5 --raid-devices=3 /dev/hdc /dev/hdd missing`
- On crée un array RAID 5 /dev/md1, avec 3 slots de disques, mais seulement 2 disques `/dev/hdc` et `/dev/hdd`
- L'array fonctionne quand même : le 3ème disque est reconstitué virtuellement à partir des 2 autres
- Quand on ajoutera le 3ème disque, il sera resynchronisé par rapport aux données des 2 autres

Modifier un array

Ajouter/retirer des disques

- `mdadm /dev/md1 -f /dev/hdc -r /dev/hdc -a /dev/hde`
- On marque le disque `/dev/hdc` comme faulty (`-f`), on le retire (`-r`), et on ajoute `/dev/hde` (`-a`)

Modifier la taille d'un array

- `mdadm --grow /dev/md1 --raid-devices=6`
- On modifie un array `/dev/md1` pour passer à 6 slots de disques
- Ça ne marche pas si `/dev/md1` est utilisé et que l'opération le rendra inconsistent (exemple : RAID 5 avec 3 disques à l'origine)

Autres commandes utiles

Regarder l'état d'un array

```
mdadm --detail /dev/md1
```

Passer un array un read-write

(Lance la resynchronisation par exemple)

```
mdadm --readwrite /dev/md1
```

Présentation

Gros LVM over RAID exportant ses disques en iSCSI
Connecté en gigabit aux serveurs (fx, fy, ... ?) via un switch séparé du reste du réseau

iSCSI

iSCSI

- iSCSI : Internet SCSI
- Porte le protocole SCSI sur une connexion TCP/IP
- En pratique, permet d'avoir un contrôle direct et complet d'un block device, exactement comme s'il était sur la machine
- Cartes iSCSI permettant à un ordinateur de booter directement sur un disque iSCSI

Interface telnet

Interface telnet

- Slon (la baie de disque) est accessible par telnet

Slonlib

Slonlib

- Dans `/usr/scripts/gestion/iscsi/`, divers scripts de gestion des disques iSCSI
- `slonlib.py` : librairie d'abstraction de la baie de disques
- Association automatique des domU à leurs disques iSCSI respectifs
- Créer un disque : `slon.create_volume(name="abc", size=10)` (avec `slon=Slon()`)
- Supprimer un disque : `slon.delete_volume("abc")`
- Envoyer des commandes arbitraires : `slon().cmd("help")`

Slonlib

Mappings

- Les disques iSCSI sont récupérés sur le dom0 via un client iSCSI (qui crée des block devices `/dev/sdX`)
- Un fichier de mapping (`/usr/scripts/var/iscsi_names.py`), généré par `slon-get-volume-mapping.py`, récupère la correpondance lun <-> nom sur la baie de disques
- `udev-get-iscsi-name.py` permet de regarder à quel volume (quel nom) correspond chaque `/dev/sdX` (en regardant le lun associé à chacun)
- Les symlinks `/dev/iscsi_nom` pointent après vers les différents disques
- Les domU utilisent ensuite `/dev/iscsi_nom-du-domU_nom-du-disque` comme disques

Merci de m'avoir écouté !

Des questions ?