Raid logiciel (mdadm)

Sommaire

Présentation et avertissements

Mdadm est une solution de RAID logiciel très fiable et très répandue dans la communauté.
Visant une configuration de type RAID1, ce tutoriel peut s'adapter à d'autres configurations (RAID5 et autres) à condition de jeter un oeil aux sources de cet article que vous trouverez dans les liens suivants :
[EN]https://raid.wiki.kernel.org/index.php/Linux_Raid
[EN]http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch26_:_Linux_Software
[EN]http://linuxdevcenter.com/pub/a/linux/2002/12/05/RAID.html
Désolé, toutes ces sources sont en anglais, mais les sources françaises ne m'ont pas plu ; trop souvent de simples billets qui balancent des commandes toutes les deux lignes sans trop expliquer pourquoi...

Je me suis efforcé de détailler toutes les étapes requises pour construire une grappe RAID1 avec mdadm, et je traite uniquement de ce cas précis ! (même si j'en rajoute un peu à la fin...). Les plus érudits d'entre vous trouveront ce tutoriel bien long pour pas grand chose mais c'est tout à fait volontaire ; l'objectif étant d'apporter une source française supplémentaire sur le sujet, et qu'elle soit plus abordable aux novices que celles déjà présentes actuellement.

La manipulation des disques dur restera toujours une opération délicate demandant un maximum d'attention et ne devant pas se faire à la va-vite. Rien de mieux qu'un bon dimanche pluvieux pour s'aventurer dans ces contrées. Ne jamais oublier que les erreurs de manipulation sur les disques sont souvent irréversibles. Vous l'aurez compris, je vous dis tout ça afin de vous désigner comme seul responsable en cas de problèmes qui pourraient survenir sur votre matériel.
Aussi, si les termes RAID1, RAID5, array, etc... ne vous disent rien, merci de vous documenter un minimum sur cette technologie. Par exemple :
[FR]http://fr.wikipedia.org/wiki/RAID_%28informatique%29#RAID_0_:_volume_agr.C3.A9g.C3.A9_p

[FR]http://www.labo-linux.org/cours/module-1/chapitre-13-raid-et-lvm/
[FR]http://www.pcworld.fr/article/materiel/stockage/le-raid-theorie-et-pratique/116001/

Architecture utilisée

Info système : Debian Lenny - kernel 2.6.26-2 i686 - mdadm v2.6.7.2

Ne voulant pas me lancer dans l'aventure périlleuse d'une description détaillée de toutes les configurations RAID possibles et imaginables, cet article se concentre simplement sur un cas précis en guise d'exemple concret d'utilisation : il s'agit ici d'une configuration loin d'être exotique puisqu'elle ne met en scène que **2 disques identiques assemblés en RAID1 et installés sur un système Debian déjà existant**. Cette grappe Raid ne contiendra donc pas le système. Elle sera juste destinée à recevoir des données et profiter de la tolérance de panne apportée par le Raid 1 appelé aussi "Disque en miroir"

Prenons 2 disques dur SATA identiques de 1To chacun. Une fois installés et notre Debian redémarrée, ils sont identifiables avec la commande # fdisk -l. Dans mon cas, ils répondent aux doux noms de /dev/sdb et /dev/sdc.

Deux alternatives se présentent alors :

- Soit vos disques dur sont déjà partitionnés voir même formatés avec un système de fichiers (ext3, reiserfs, ntfs, etc..)

Pour le savoir, observez en détail la sortie de la commande # fdisk -l. Chaque description de disque commence par les lignes suivantes (ex : pour /dev/sdb):

```
Disk /dev/sdb: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000b7bff
```

Si il y a une ou des partitions, on trouvera un peu plus bas un tableau de ce type :

```
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        5496    44146588+  83  Linux
/dev/sdb2            5497      121601   932613412+   7  HPFS/NTFS
```

Chaque /dev/sdbX désignant donc une partition.

- Soit la table des partitions est vierge comme dans le cas d'un disque neuf

Si vos disques sont vierges ou neuf, il y a de grande chance que vous n'ayez rien du tout dans le tableau cité ci-dessus. Seule la première ligne apparaîtra comme ceci :

```
Device Boot      Start         End      Blocks   Id  System
```

Il est possible aussi que même cette ligne ne soit pas présente avec à la place ce genre de commentaire :

```
Disk /dev/sdb doesn't contain a valid partition table
```

Préparation des disques (fdisk)

Attaquons nous maintenant à fdisk dans sa fonctionnalité d'éditeur de table des partitions. Comprenez bien pour les débutants qu'il s'agit ici de modifier la table des partitions, en aucun cas il n'est question de système de fichiers et de formatage.

Avertissement : dans ce qui suit, nous allons toucher à la table des partitions de vos deux disques, si vous continuez, vous acceptez alors que les données de vos deux disques ne seront plus disponibles (sauf si vous restaurez la table...). Ces deux disques seront intégralement utilisés par le RAID1

Fdisk : Modification de la table des partitions du premier disque

Si il y a une ou des partitions sur vos disques, alors elles contiennent peut être aussi un système de fichiers qui est déjà monté dans le système. Assurez vous donc de démonter ces partitions. Exemple pour /dev/sdb1 et /dev/sdb2

```
# umount /dev/sdb1
# umount /dev/sdb2
```

Exécutons maintenant fdisk pour le disque /dev/sdb :

```
# fdisk /dev/sdb
.
Command (m for help):
```

Ne soyons pas trop téméraire et appuyons sur **m** :

```
Command action
  a   toggle a bootable flag
  b   edit bsd disklabel
  c   toggle the dos compatibility flag
  d   delete a partition
  l   list known partition types
  m   print this menu
  n   add a new partition
  o   create a new empty DOS partition table
  p   print the partition table
  q   quit without saving changes
  s   create a new empty Sun disklabel
  t   change a partition's system id
  u   change display/entry units
  v   verify the partition table
  w   write table to disk and exit
  x   extra functionality (experts only)
```

Nous voilà prêt à faire mumuse. Tant que vous ne pressez pas la touche **w** (Ecrire la table sur le disque et quitter) alors vous ne risquez pas grand chose. Et si vous ne savez plus ce que vous faites alors appuyez sur **q** (quitter sans sauvegarder) ou bien arrêtez la picole... Maintenant je vous propose un truc : on va

effacer la table des partitions pour ensuite en créer une nouvelle toute fraîche. Ce sera moins fastidieux que de modifier à la main les entrées de chaque table.
Tapez donc la lettre **p** (print) afin d'afficher votre table, ex :

```
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        5496    44146588+  83  Linux
/dev/sdb2            5497      121601   932613412+   7  HPFS/NTFS
```

Si votre table est vide, alors passez directement à la partie suivante **Fdisk : Création de la table des partitions sur le premier disque**
Pour les autres, tapez **d** (delete), on vous demande alors le numéro de partition, tapez **2**. Nous venons donc de choisir la suppression de /dev/sdb2. Si vous refaites **p**, /dev/sdb2 a disparu. Ressaisissez à nouveau la lettre **d** ce qui aura pour effet de supprimer la dernière partition sans avoir à confirmer le numéro de celle-ci. Refaites **p**... Plus de partoch !
A présent, appuyez sur **w** pour valider les changements et supprimer les partitions (ici un redémarrage complet peut être nécessaire)

Fdisk : Création de la table des partitions sur le premier disque

Quelques explications s'imposent : vous n'êtes peut être pas sans savoir que dans la table des partitions il existe une information qui caractérise le type de partition déclaré. Il s'agit d'un ID représenté par un code hexadécimal de 1 octet. Pour visualiser la liste de ces ID disponibles, utilisez fdisk sur un disque de votre choix contenant une ou des partitions (ex : fdisk /dev/sda). Puis tapez **t** et choisissez le numéro de la partition à modifier **1-4**. Taper **L** pour visualiser la liste des ID ; on remarque alors que suivant l'OS utilisé et le futur système de fichiers installé on choisira un code différent. Vous pouvez mettre ce que vous voulez puis appuyez sur **p** pour observer le changement (les deux derniers champs ont changé : *Id* et *System*).
Une fois retourné dans l'invite de commande principal de fdisk, tapez bien **q** pour quitter sans sauvegarder

Mais retournons à nos manchots... Après avoir effacé la table des partitions de notre premier disque /dev/sdb, nous allons créer une nouvelle entrée dans cette table qui désignera une partition primaire occupant l'intégralité du disque :

```
# fdisk /dev/sdb
```

Dans fdisk tapez **n** pour ajouter une nouvelle partition

```
Command action
  e   extended
  p   primary partition (1-4)
```

Tapez **p** pour créer une partition primaire

```
Partition number (1-4):
```

Entrez le numéro 1 pour créer /dev/sdb1. Fdisk pose ensuite 2 autres questions auxquelles vous pouvez laisser les valeurs par défaut ; il veut connaître la valeur du premier cylindre (First cylinder...) et celle du dernier (Last cylinder...), il s'agit ni plus ni moins de l'espace alloué à cette partition. En laissant par

défaut, c'est tout l'espace disque qui sera utilisé.
De retour à notre invite de commande fdisk, nous pouvons appuyer sur **p** pour admirer notre nouvelle partition :

```
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1      121601   976760001   83  Linux
```

Comme vu précédemment, nous remarquons les champs Id et System soit 83 Linux ; il s'agit du type par défaut.
Passons alors le type de partition en "FD Linux RAID autodetect". Taper **t** et entrez les caractères fd pour désigner ce type de partition. Vous pouvez vérifier le changement avec **p**.

Note : si vous laissez le type 83 Linux ça fonctionnera aussi très bien car mdadm écrit des "md superblock" sur les partitions (ou disques) lors de la création des ensembles. Ces "md superblock" donnent de précieuses informations sur l'intégrité et la constitution des grappes. Mdadm s'en sert pour identifier les partitions (ou disques) faisant partie d'un RAID. Je voulais au départ réunir suffisamment d'informations afin d'expliquer le choix d'un type FD ou 83 mais je n'ai rien trouvé de convaincant. Si des experts parmi nous se sentent capable d'expliquer le pourquoi du comment c'est pas de refus.

Terminer en tapant **w** pour enregistrer les modifications.
Voila, notre premier disque /dev/sdb est prêt pour l'intégration au RAID logiciel de mdadm.

Préparons le deuxième disque

- Soit vous refaites toutes les manip depuis le début de la partie **3 Préparations des disques (fdisk)** en l'appliquant à /dev/sdc

- Soit vous tapez simplement la commande qui suit. Celle-ci a pour effet de recopier fidèlement la table des partitions de /dev/sdb sur /dev/sdc :
/!\Attention/!\ NE TAPEZ PAS CETTE COMMANDE SANS VÉRIFIER A DEUX FOIS LE NOM DES DISQUES !! /dev/sdb et /dev/sdc sont ici des exemples, vous aurez compris j'espère que ce n'est peut être pas votre cas. Cette commande ne demande aucune confirmation !

```
# sfdisk -d /dev/sdb | sfdisk /dev/sdc
```

On vérifie que nos deux disques présentent les même partitions :

```
# fdisk -l
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1      121601   976760001   fd  Linux raid autodetect
...
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1               1      121601   976760001   fd  Linux raid autodetect
...
```

Les disques sont ready to RAID ! Passons à mdadm...

Création du RAID1 (mdadm)

Installation et préparation (mdadm):

```
# aptitude install mdadm
```

Pendant l'installation, une fenêtre de dialogue vous demande si des ensembles RAID doivent être démarrés juste avant la procédure de démarrage système. Cette fonctionnalité est utile si votre système est contenu dans un ensemble RAID. Ces ensembles portent des noms du type md0, md1, md2, etc... Le choix par défaut est "all" qui démarre tous les mdX avant le système. Nous pouvons laisser ce paramètre par défaut même si notre système n'est pas localisé sur un ensemble RAID.

Il est temps de créer notre ensemble RAID qui se nommera /dev/md0

```
# mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

On observe donc les paramètres --level=1 pour RAID1 et --raid-devices=2 pour deux partitions (ou deux disques)
Ceci peut se condenser en : (c'est exactement la même commande !)

```
# mdadm -Cv /dev/md0 -l1 -n2 /dev/sdb1 /dev/sdc1
```

Note : il est possible d'intégrer à notre RAID1 un disque de spare qui sera donc automatiquement synchronisé en cas de panne d'un disque du miroir. Pour cela, il suffit de préparer une troisième partition sur un troisième disque (ex : /dev/sdd1) et de créer l'ensemble RAID /dev/md0 avec la commande suivante :

```
# mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1 --spare-devices=1 /dev/sdd1
```

Notre ensemble RAID est alors en phase de construction, vous pouvez observer son état d'avancement avec la commande :

```
# cat /proc/mdstat
```

Qui devrait vous donner quelque chose du genre :

```
Personalities : [raid1]
md0 : active raid1 sdb1[0] sdc1[1]
      976759936 blocks [2/2] [UU]
      [===============>.....]  resync = 78.4% (765779789/976759936) finish=14.2min speed=312K/sec
```

Cela peut être relativement long suivant la taille des partitions. Pour un RAID1 de 2x1To j'ai attendu plus d'1h.
Tant que cette même commande ne nous retourne pas ceci :

```
Personalities : [raid1]
md0 : active raid1 sdb1[0] sdc1[1]
      976759936 blocks [2/2] [UU]
```

```
.
unused devices: <none>
```

...c'est que le RAID n'est pas encore prêt. Patientez donc un peu... et passez à l'étape suivante.

Vous pouvez aussi suivre la construction de l'ensemble Raid "en direct" avec la commande suivante :

```
# watch -n 1 cat /proc/mdstat
```

Formatage de l'array /dev/md0

Notre ensemble RAID1 (array) est matérialisé au niveau système par le fichier spécial de type bloc /dev/md0. Nous pouvons manipuler /dev/md0 comme nous le ferions avec une partition classique (ex : /dev/hda1 ou /dev/sdb3, etc..) c'est pourquoi nous allons tout de suite y créer un système de fichiers (exemple avec ext3)

```
# mkfs.ext3 /dev/md0
```

*Note : ajuster l'option **-m** à la commande ci dessus si vous voulez diminuer l'espace réservé au super-user (cf. man mkfs.ext3). Par défaut il est de 5%, et sur 1To ça fait déjà un paquet de giga en moins.*

On peut désormais monter son /dev/md0 et le rendre permanent dans le /etc/fstab.

Note pour ceux qui se demandent comment qu'on fait : pour monter de façon permanente un système de fichiers ext3 et le rendre accessible en lecture/écriture par les utilisateurs standards du système, faites ceci :

- Créez un nouveau groupe utilisateur, il s'agit de ceux qui utiliseront le RAID. Appelons le... je sais pas moi... raid

```
# addgroup raid
```

- Puis, pour chaque utilisateur que l'on souhaite intégrer au groupe raid, exécutons la commande suivante : (exemple avec toto)

```
# adduser toto raid
```

- Créez le point de montage là ou vous voulez : (exemple dans /media/raid1)

```
# mkdir /media/raid1
```

- Editez le fichier /etc/fstab et rajoutez-y cette ligne

```
/dev/md0   /media/raid1   ext3      defaults      0   0
```

- Il est temps de monter notre chère partoch

```
# mount /dev/md0
```

- Et enfin, appliquons les droits adéquats au point de montage :

```
# chown root:raid /media/raid1 && chmod 775 /media/raid1
```

Certains se demandent peut être : "Mais pourquoi ne pas utiliser directement les options gid,uid,umask dans le /etc/fstab au lieu de s'embêter avec des chown, chmod, et autre addgroup ?!.." C'est tout simplement parce que le système ext3 ne supporte pas de telles options. Ces options la existent seulement pour des systèmes de fichiers étrangers au monde linux qui ne gèrent pas de la même façon les permissions UNIX (comme fat, ntfs,...). Or ext3 est un système de fichiers natif, qui supporte évidemment les permissions UNIX, donc on s'en sert !

Voilà, notre grappe RAID1 est maintenant prête à être utilisée, mais avant sauvegardons notre configuration dans le fichier /etc/mdadm/mdadm.conf :

```
# mdadm --detail --scan --verbose > /etc/mdadm/mdadm.conf
```

Cette commande est à renouveler après toutes modifications sur les arrays (ajout de disque, remplacement de disque HS, etc…)

Maintenance

C'est bien beau de mettre en œuvre une politique de tolérance de panne mais que fait on... en cas de panne ?

Comment détecter une panne ?

Le fichier système /proc/mdstat permet d'avoir un aperçu rapide de l'état de nos ensembles RAID.

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdb1[0] sdc1[1]
      976759936 blocks [2/2] [UU]
.
unused devices: <none>
```

Un coup d'oeil rapide sur le flag [UU] et nous savons que tout est OK. Si il y avait un problème sur un des disques nous aurions [_U] ou [U_]

Comment identifier le disque fautif ?

La ligne md0 : active raid1 sdb1[0] sdc1[1] peut apporter cette information : si une partition manque à l'appel alors nous aurons par exemple md0 : active raid1 sdc1[1] (il manque sdb1 !) ou encore ceci md0 : active raid1 sdb1[0](F) sdc1[1] (notez le (F) pour faulty !).

On peut également consulter la sortie de cette commande :

```
# mdadm --detail /dev/md0
```

Repérez la ligne State : si vous voyez ceci : ***State : clean, degraded*** ce n'est pas bon signe... l'un des disques pose problème.
A la fin, vous avez un tableau listant les composants de l'ensemble RAID. Quand tout va bien, vous voyez vos périphériques (/dev/sdb1, /dev/sdc1, etc..) précédés de la mention ***active sync***. Mais si ça va mal, on trouvera un commentaire différent, par exemple ***faulty spare***, ou ***removed***)

Reste enfin la commande système **dmesg** ou encore les fichiers journaux /var/log/syslog et /var/log/messages qui peuvent apporter des informations précieuses en cas de problème.

Comment automatiser la détection de panne ?

- Soit vous scriptez en vous basant sur le fichier /proc/mdstat ou la sortie de mdadm --detail puis vous appelez ce script via une crontab.
Pour ceux qui manqueraient d'inspiration, voici un exemple tout bête avec le célèbre Conky :
Placez cette ligne quelque part dans la partie TEXT de votre .conkyrc.

```
${texeci 3600 awk ' $2 == "blocks" && $4 != "[UU]" {print "DANGER ! Erreur RAID détectée !!!"}' /proc/mdstat}
```

Toutes les 3600s (1h) conky exécutera la commande awk qui vérifiera si le flag [UU] est bien présent. Si ce n'est pas le cas, le message "DANGER ! Erreur RAID détectée !!!" s'affichera sur votre bureau. Notez tout de même que cette commande n'est compatible que pour un seul ensemble RAID de 2 disques. Veuillez adapter cette commande si votre configuration est différente.

- Soit vous utilisez le mode monitor de mdadm.
Mdadm intègre un système d'alerte automatique avec possibilité de passer en paramètre une adresse email (--mail) ou bien un programme à exécuter (--alert) (cf. man mdadm)

Exemple :

```
# mdadm --monitor -f --mail=votre@email.com --delay=3600 /dev/md0
```

-f permet de "daemoniser" le mode monitor (comprendre exécuter en arrière plan). Mais si par malheur le processus tombe, alors vous ne serez plus averti.

Cas 1 : panne d'un disque de l'array (ex : /dev/sdb)

Une fois le disque fautif identifié, il faut signaler à mdadm que nous voulons le retirer de notre ensemble RAID.

```
# mdadm --manage /dev/md0 --remove /dev/sdb1
```

En cas de grosse panne disque, ce dernier n'est peut être plus détecté par le système. Cette commande renverra donc une erreur signalant que /dev/sdb1 n'existe pas. Ce n'est pas génant, continuons...

/!\ CAS PARTICULIER /!\ : si votre disque n'est plus détecté et que le redémarrage de la machine est impossible (contrainte production). Allez voir les précisions de AnatomicJC ICI (http://forum.debian-fr.org/viewtopic.php?f=1&t=27582#p273926)

Eteindre la machine et placer un nouveau disque de taille identique ou supérieure. Une fois redémarrée, vérifier la présence du nouveau disque avec # fdisk -l et recommencer les étapes de la partie **3 Préparations des disques (fdisk)**. Vous pouvez aussi recopier simplement la table des partitions de /dev/sdc sur /dev/sdb comme nous l'avons vu dans la partie **3.3 Préparons le deuxième disque**...

/!\Attention/!\ NE TAPER PAS CETTE COMMANDE SANS VÉRIFIER A DEUX FOIS LE NOM DES DISQUES !!

```
# sfdisk -d /dev/sdc | sfdisk /dev/sdb
```

Ajoutons le disque à l'ensemble dégradé :

```
# mdadm --manage /dev/md0 --add /dev/sdb1
```

L'ensemble RAID est alors en reconstruction, cela peut être très long. Vous pouvez observer l'avancement en consultant le fichier /proc/mdstat. Vous remarquerez alors la mention "recovery" à droite de la jauge de progression.

Une fois la reconstruction terminée, ne pas oublier de mettre à jour le fichier de configuration :

```
# mdadm --detail --scan --verbose > /etc/mdadm/mdadm.conf
```

Cas 2 : panne du disque système, la grappe RAID est OK

Voici un cas un peu particulier mais qui m'est arrivé donc je tenais à en parler ici. Nous sommes donc dans le cas donné pour exemple dans cet article à savoir ; une Debian installée sur un seul disque et une grappe RAID1 pour les données construite avec mdadm. Imaginons que votre disque système tombe en rade mais que votre grappe reste OK. Il vous faut donc réinstaller votre Linux avec mdadm et lui faire comprendre qu'une array existe déjà.

Dans l'ordre de préférence faisons ceci :

Choix 1 - Si tout va bien, lors de la nouvelle installation de mdadm, il va détecter les "md superblock" et en déduire donc qu'une grappe RAID1 existe entre les partitions /dev/sdb1 et /dev/sdc1. Surveillez donc votre /proc/mdstat et attendez que la synchro soit terminée avant de redémarrer votre machine.

Choix 2 - Si vous aviez sauvegardé le fichier /etc/mdadm/mdadm.conf, vous pouvez le restaurer et taper la commande suivante :

```
# mdadm --assemble --scan
```

Ceci à pour effet de scanner le mdadm.conf et d'assembler toutes les arrays y figurant. En sachant qu'un simple redémarrage de la machine avec le mdadm.conf fraîchement restauré suffit à offrir le même résultat.

Choix 3 - S'il ne se passe rien et que vous n'avez plus votre mdadm.conf alors il vous faut connaître le nom de votre array et les partitions qui la composent. Dans notre cas c'est pas bien dur ; nos deux partitions /dev/sdb1 et /dev/sdc1 sont en RAID1 dans l'array /dev/md0. On tape donc :

```
# mdadm --assemble /dev/md0 /dev/sdb1 /dev/sdc1
```

Choix 4 - Si l'architecture est plus complexe et que vous pataugez dans la semoule, il est judicieux de vous aider des commandes # *fdisk -l* et # *mdadm –-examine /dev/<partition>* pour identifier le bouzin.

Exemple :

```
# fdisk -l
...
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1      121601   976760001   fd  Linux raid autodetect
...
...
```

Je vois ici que /dev/sdb1 fait certainement partie d'un ensemble RAID. Alors j'enquête....

```
# mdadm --examine /dev/sdb1
mdadm: metadata format 00.90 unknown, ignored.
/dev/sdb1:
          Magic : a92b4efc
        Version : 00.90.00
           UUID : 3b2be7cf:1eca6c08:a962df05:773a6f64
  Creation Time : Sun Dec 14 20:41:21 2008
     Raid Level : raid1
  Used Dev Size : 976759936 (931.51 GiB 1000.20 GB)
     Array Size : 976759936 (931.51 GiB 1000.20 GB)
   Raid Devices : 2
  Total Devices : 2
Preferred Minor : 0
.
    Update Time : Fri May  7 17:06:49 2010
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0
       Checksum : f320b0b5 - correct
         Events : 556
.
.
      Number   Major   Minor   RaidDevice State
this     1       8       17        1      active sync   /dev/sdb1
.
   0     0       8        1        0      active sync   /dev/sdc1
   1     1       8       17        1      active sync   /dev/sdb1
```

Grâce à ces infos, je sais donc que la partition /dev/sdb1 fait partie d'un ensemble RAID1 et que sa compagne est la partition /dev/sdc1 (voir à la fin)
On tape donc :

```
# mdadm --assemble /dev/md0 /dev/sdb1 /dev/sdc1
```

Une fois que tout est en ordre et bien synchro, tapez ceci :

```
# mdadm --detail --scan --verbose > /etc/mdadm/mdadm.conf
```

Aller plus loin...

En guise de conclusion, je voudrais donner quelques pistes pour aller plus loin :

Beaucoup d'entre vous voudront certainement mettre en place un miroir sur leur système plutôt que sur les données. On retrouve d'ailleurs souvent sur des serveurs une configuration du type :
- 2 disques en RAID1 pour le système
- 3(ou+) disques en RAID5 pour les données

RAID1 système : il est possible d'utiliser l'installeur debian pour mettre en place une architecture de type RAID1+LVM sur le système. Cependant, la partition /boot doit être en dehors du LVM à cause d'une limitation de GRUB premier du nom. GRUB2 supporte maintenant le boot sur LVM (avis aux experts pour compléments d'infos, merci).

Avec notre cher collaborateur lol ;) nous avons testé ceci grâce à ces explications :
http://blog.le7.net/linux/debian/installation-debian-ubuntu-avec-raid-1/
Merci en passant à son auteur. Ce tutoriel fonctionne au poil avec une install lenny, j'ai effectué également des tests de panne et de récupération avec succès.

RAID5 de données : sur la question du stockage des données, il est plus intéressant sur un rapport prix/capacité d'opter pour un RAID5 plutôt qu'un RAID1. En effet, un RAID1 offre une capacité de stockage correspondant à la moitié de la taille totale des deux disques, alors qu'avec un RAID5 à 3 disques, nous avons les 2/3 de la taille totale des 3 disques.

Exemple : prenons des disques de 1To à 100 euros
RAID1 = 200 euros pour 1To de données = 0.2 euro / Go
RAID5 = 300 euros pour 2To de données = 0.15 euro / Go
La mise en place d'un RAID5 se fait sensiblement de la même manière qu'un RAID1. Evidemment, la commande mdadm de création de l'array sera différente :

Exemple :

```
# mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sdb1 /dev/sdc1 /dev/sdd1
```

Cependant, je tiens à signaler qu'on trouve des personnes se plaignant des performances RAID5 avec mdadm. Je vous conseille donc vivement de jeter un oeil à la fin de cet article qui est en anglais mais se comprend assez facilement : https://raid.wiki.kernel.org/index.php/RAID_setup

En bref, voici les paramètres qui permettraient d'offrir des performances dites "normales" dans la plupart

des cas pour un RAID5 à 3 disques :

- Construire son RAID5 en positionnant la taille du chunk à 128KB :

```
# mdadm --create --verbose /dev/md0 --level=5 --chunk=128 --raid-devices=3 /dev/sdb1 /dev/sdc1 /dev/sdd1
```

- Choisir la taille des blocs du système de fichiers résidant sur /dev/md0 : prenons par exemple 4KB qui est une valeur adaptée pour l'utilisation de gros fichiers.

- La taille du chunk et des blocs sur le système de fichiers vont déterminer deux autres paramètres : stride et stripe-width
stride = chunk / blocs = 128 / 4 = 32KB
stripe-width = stride x (nbr disques - 1) = 32 x (3 - 1) = 64KB

- Créons le système de fichiers avec ces valeurs :

```
# mkfs.ext3 -v -b 4096 -E stride=32,stripe-width=64 /dev/md0
```

Enfin, d'après cet article (http://peterkieser.com/2009/11/29/raid-mdraid-stripe_cache_size-vs-write-transfer/), nous pouvons jouer sur la valeur de stripe_cache_size. La valeur optimale semble être 8192 :
/!\ Attention, cette commande concerne l'ensemble md0 ! Adapter en conséquence...

```
# echo 8192 > /sys/block/md0/md/stripe_cache_size
```

Rien d'objectif dans ces choix, alors faites vos tests et venez nous en faire part sur le forum (http://debian-fr.xyz) !

Spatule 18 mai 2010 à 09:40 (UTC)

Récupérée de « https://wiki.debian-fr.xyz/index.php?title=Raid_logiciel_(mdadm)&oldid=6965 »
