NBD : partagez les “Block Devices” via le réseau

Lorsqu'un utilisateur travaille sur un terminal X, tout se passe comme s'il était directement sur le serveur. Donc, s'il tente d'utiliser le lecteur de disquettes par `/dev/fd0`, il accèdera à celui du serveur, et non à celui éventuellement présent sur le terminal... ce qui est loin d'être idéal. Voyons comment résoudre ce problème.

Si vous avez lu l'article sur l'installation d'un terminal X, une question s'est peut-être imposée à vous : comment accéder par exemple au lecteur de disquette du terminal ? NBD peut apporter une solution à ce problème. NBD est l'acronyme de Network Block Device. Il permet à une machine d'accéder via un réseau à certaines ressources matérielles d'une autre machine. Je vous propose de découvrir ce mécanisme.

Présentation et installation

NBD est composé de trois éléments :

- Un module à charger dans le noyau Linux (ce module est normalement présent dans tous les noyaux depuis le 2.1.101) ;
- Un programme serveur `nbd-server`, donc à utiliser sur le serveur mettant ses ressources à disposition ;
- Un programme client `nbd-client`, donc à utiliser sur le client profitant des ressources du serveur.

Les grandes distributions proposent généralement au moins un paquetage contenant tout ce dont vous avez besoin, et le noyau pré-compilé vient accompagné du module susmentionné. Si vous voulez installer ces éléments vous-même, le site de NBD [1] vous donne accès à une archive contenant les programmes client et serveur. Il vous faudra éventuellement recompiler votre noyau, pour inclure le support NBD. Il s'agit de l'option `CONFIG_BLK_DEV_NBD`, dénommée “Network block device support” dans la section “Block devices”. Le noyau doit naturellement être configuré de la sorte côté client et côté serveur. Le serveur peut ensuite se contenter du programme `nbd-serveur`, le client du programme `nbd-client`.

Utilisation

Une fois tous les composants installés sur vos systèmes, passons à la pratique.

Côté serveur

La ligne de commande pour le programme serveur est (étant root) :

```
# modprobe nbd
# nbd-server <port> <fichier> [-r] [-c] [-a <timeout>]
```

La première ligne n'est utile que si vous n'avez pas inclus NBD directement dans le noyau, mais plutôt que vous l'ayez compilé sous forme de module. La seconde démarre le serveur, les principaux arguments étant :

- `<port>` est le port IP sur lequel le serveur va attendre les connexions distantes ;
- `<fichier>` est le fichier qui sera “exporté” par le serveur ; cela peut être un fichier régulier (qui devra être créé à la taille voulue), ou bien un fichier spécial correspondant à un périphérique fonctionnant par blocs, comme une partition de disque dur ;
- `-r` (optionnel) indique que l'accès sera forcément en lecture seule ;
- `-c` (optionnel), utilisé dans les accès en lecture/écriture, permet de ne pas effectuer les opérations d'écriture sur le fichier exporté, mais dans un autre fichier (régulier) ; ce fichier sera détruit lors de la fermeture de la connexion. Cette option ralentit les accès, mais permet par exemple de simuler une écriture sur un CD-ROM ;
- `-a <timeout>` (optionnel) provoque l'arrêt du serveur si une connexion est inactive pendant une certaine durée, donnée en secondes ; cela évite d'avoir un processus dormant pour l'éternité, aussi l'utilisation de cette option est-elle vivement recommandée.

Par exemple, supposons que je veuille donner l'accès en lecture/écriture à la partition `hda7` de mon disque dur, par le port 17788. La commande à lancer sera alors :

```
# nbd-server 17788 /dev/hda7
```

Côté client

Le client quant à lui utilisera la commande :

```
# modprobe nbd
# nbd-client [bs=<blocksize>] <serveur> <port> <périphérique nbd>
[-swap]
```

Comme précédemment, la première ligne dépend de votre configuration. Les paramètres de cette simple commande sont :

- `bs=<blocksize>` (optionnel) indique la taille des blocs à utiliser, par défaut 1024 ; cette taille n'a pas besoin de correspondre à la taille réelle des blocs du périphérique

auquel on accède. Les valeurs possibles sont 512, 1024, 2048 et 4096 ;

- `<serveur>` est le nom ou l'adresse IP de la machine sur laquelle fonctionne le serveur ;
- `<port>` est le port auquel se connecter ;
- `<périphérique nbd>` désigne le fichier spécial qui sera utilisé localement, j'y reviens dans un instant ;
- `-swap` (optionnel) indique que le périphérique NBD sera utilisé en tant que mémoire virtuelle (swap).

Du point de vue du client, la communication s'effectue au travers d'un fichier spécial, de type blocs, de numéro majeur 43 et de numéro mineur compris entre 0 et 128. La première chose à faire est donc de créer un tel fichier spécial, où bon vous semble, par exemple :

```
# mknod nd0 b 43 0
```

Cela étant fait, nous pouvons nous connecter au serveur. Supposons que celui lancé plus haut l'a été sur une machine nommée `tx01`, la commande à utiliser est alors :

```
# ndb-client tx01 17788 nd0
```

Nous pouvons maintenant utiliser le fichier spécial `nd0` comme n'importe quel périphérique de type blocs ! Par exemple, créer un système de fichiers dessus :

```
# mke2fs nd0
```

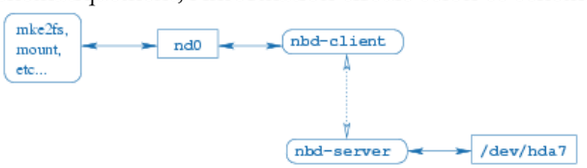
...puis monter ce système de fichiers, tout à fait normalement :

```
# mount nd0 /mnt
```

Lorsque vous n'avez plus besoin de la connexion, après avoir convenablement démonté les éventuels systèmes de fichiers, vous pouvez couper la communication avec la commande :

```
# nbd-client -d nd0
```

Schématiquement, l'information circule selon ce schéma,



le lien pointillé représentant la circulation sur le réseau :

Si vous envisagez d'utiliser NBD pour *swapper* au travers d'un réseau, pensez à utiliser l'option `-swap` : sans elle vous risquez de rencontrer des problèmes d'inter-blocages.

Nous avons donc apparemment une solution à notre problème d'accès aux unités d'un terminal X : chaque terminal lance un serveur NBD par périphérique auquel on veut accorder l'accès, le serveur principal lançant autant de clients que nécessaire. Les périphériques NBD seraient alors créés dans les répertoires des utilisateurs. Dans ce cas, il peut être intéressant d'utiliser un tunnel SSH pour crypter, et surtout compresser les données en transit.

Problème : les périphériques amovibles

Lors de mes expérimentations, je suis tombé sur un problème lié aux périphériques amovibles. Supposons que l'on démarre un serveur NBD sur un lecteur de disquettes vide. Le client peut se connecter, mais si on tente d'utiliser le pseudo-périphérique NBD après avoir inséré une disquette, on obtient un message d'erreur, comme si le lecteur était toujours vide. J'ai finalement constaté qu'il était nécessaire que la disquette soit dans le lecteur quand le client se connecte ! Le même phénomène se présente pour un lecteur de CD-ROM. Chose importante, il n'est heureusement pas nécessaire de redémarrer le serveur NBD lors d'un changement dans le lecteur. Une solution possible consiste à créer un script ayant l'attribut `suid`, et appartenant à root, dans le style de celui-ci :

```
#!/bin/bash
host='echo $DISPLAY | cut -d " :" -f 1 -s -'
if [ $host == "" ]; then
  host="localhost"
fi
nbd-client $host 17788 $HOME/dev/nd0 && \
mount $HOME/dev/nd0 $HOME/floppy
```

La première ligne permet de récupérer à partir de la variable `DISPLAY` le nom de la machine faisant tourner le terminal X (ce n'est probablement pas optimisé, je ne suis pas un champion du bash !). Ce nom est stocké dans `host`, par défaut `localhost` s'il est vide. Puis nous réalisons la connexion avec le serveur NBD, le périphérique `nd0` ayant été préalablement créé dans un répertoire `dev` du répertoire de l'utilisateur. Si cette connexion réussit, le système de fichiers présent sur la disquette est monté pour y accéder. Pour que cette commande `mount` réussisse, une ligne adaptée doit se trouver dans le fichier `/etc/fstab`. À l'inverse, le script suivant déconnecte le périphérique préalablement connecté :

```
#!/bin/bash
umount $HOME/floppy && \
nbd-client -d $HOME/dev/nd0
```

Conclusion

L'utilisation de NBD est assez simple, et devrait vous donner des idées, même si le système n'est pas parfait. Il pourrait être intéressant d'avoir une option particulière permettant au client d'interroger le serveur sur l'état du périphérique, pour éviter de devoir le relancer lorsqu'on accède à un périphérique amovible. Dernier mot, il pourrait être amusant de disposer du même genre d'outil pour l'accès via réseau aux périphériques en mode caractères ! Imaginez partager par un réseau une portion de mémoire, ou un port série...

Yves Bailly
http ://www.kafka-fr.net

Références

[1] Site de NBD : http ://nbd.sourceforge.net

[2] Site originel : http://atrey.karlin.mff.cuni.cz/~pavel/nbd/nbd.html