Introduction au réseau de stockage iSCSI

Philippe Latu
philippe.latu(at)linux-france.org

http://www.linux-france.org/prj/inetdoc/

Historique des versions		
$Revision: 1633 $	$Date: 2011-04-26 11:29:04 +0200 (mar. 26 avril 2011) $	$Author: latu $
Année universitaire 2010-2011		

Résumé

Ce support de travaux pratiques est consacré à l'étude d'une infrastructure de stockage illustrant les technologies DAS (Direct Attached Storage), SAN (Storage Area Network) et la redondance de niveau 1 (RAID1). La technologie iSCSI est utilisée pour la partie SAN, c'est un exemple d'accès «en mode bloc» aux unités de stockage réseau. La redondance de niveau 1 utilise les fonctions intégrées au noyau Linux. L'infrastructure proposée montre aussi comment les différentes technologies élémentaires peuvent être combinées pour atteindre les objectifs de haute disponibilité et de sauvegarde.

Table des matières

1. Copyright et Licence 1
    1.1. Méta-information 2
    1.2. Conventions typographiques 2
2. Adressage IP des postes de travail 2
3. Technologie iSCSI et topologie de travaux pratiques 3
    3.1. Bases de la technologie iSCSI 3
    3.2. Infrastructure de stockage étudiée 4
4. Préparation d'une unité de stockage 4
    4.1. Destruction de la table des partitions 5
    4.2. Création de la table des partitions et formatage 5
5. Configuration du système initiator 7
    5.1. Sélection du paquet et lancement du service 7
    5.2. Tests de fonctionnement du service 8
    5.3. Configuration système 9
6. Configuration du système target 10
    6.1. Sélection des paquets et compilation du module 10
    6.2. Configuration du service 11
7. Configuration d'une unité logique RAID1 13
    7.1. Sélection du paquet et création de l'unité de stockage 13
    7.2. Manipulations sur l'unité de stockage RAID1 13
8. Manipulations sur machines virtuelles 14
    8.1. Préparation du système hôte 14
    8.2. Serveur target iSCSI 15
    8.3. Clients initiator iSCSI 16
9. Documents de référence 16
A. Configuration du système target sans service DKMS 17

1. Copyright et Licence

1.1. Méta-information

Cet article est écrit avec *DocBook*[1] XML sur un système *Debian GNU/Linux*[2]. Il est disponible en version imprimable au format PDF : `admin.reseau.iscsi.pdf`[3].

1.2. Conventions typographiques

Tous les exemples d'exécution des commandes sont précédés d'une invite utilisateur ou *prompt* spécifique au niveau des droits utilisateurs nécessaires sur le système.

- Toute commande précédée de l'invite `$` ne nécessite aucun privilège particulier et peut être utilisée au niveau utilisateur simple.
- Toute commande précédée de l'invite # nécessite les privilèges du super-utilisateur.

2. Adressage IP des postes de travail

À partir de l'infrastructure proposée dans la section suivante, on constitue des couples de postes de travail qui vont partager le même domaine de diffusion durant la séance de travaux pratiques.

Ces opérations de réaffectation du plan d'adressage IP sont répétées à chaque début de séance de travaux pratiques. Elles s'appuient sur les indications données dans le document *Architecture réseau des travaux pratiques*[4].

Tableau 1. Affectation des adresses et des réseaux IP

Poste 1	Poste 2	Passerelle par défaut
alderaan	bespin	`10.4.4.1/23`
centares	coruscant	`192.168.109.1/25`
dagobah	endor	`10.0.117.1/27`
felucia	geonosis	`10.7.10.1/23`
hoth	mustafar	`172.19.112.1/26`
naboo	tatooine	`192.168.111.1/25`

Une fois la passerelle du réseau IP affecté à chaque paire de postes de travaux pratiques, il faut rechercher dans le document *Architecture réseau des travaux pratiques*[5] les éléments nécessaires à la connexion physique de ces postes. Les étapes usuelles sont les suivantes :

1. Attribuer une adresse IP à chacun des postes en fonction de l'espace d'adressage du réseau défini.

2. Rechercher le numéro de VLAN correspondant au réseau IP attribué.

3. Repérer le commutateur sur lequel des ports ont été affectés au VLAN recherché. Connecter les deux postes de travaux pratiques sur les ports identifiés.

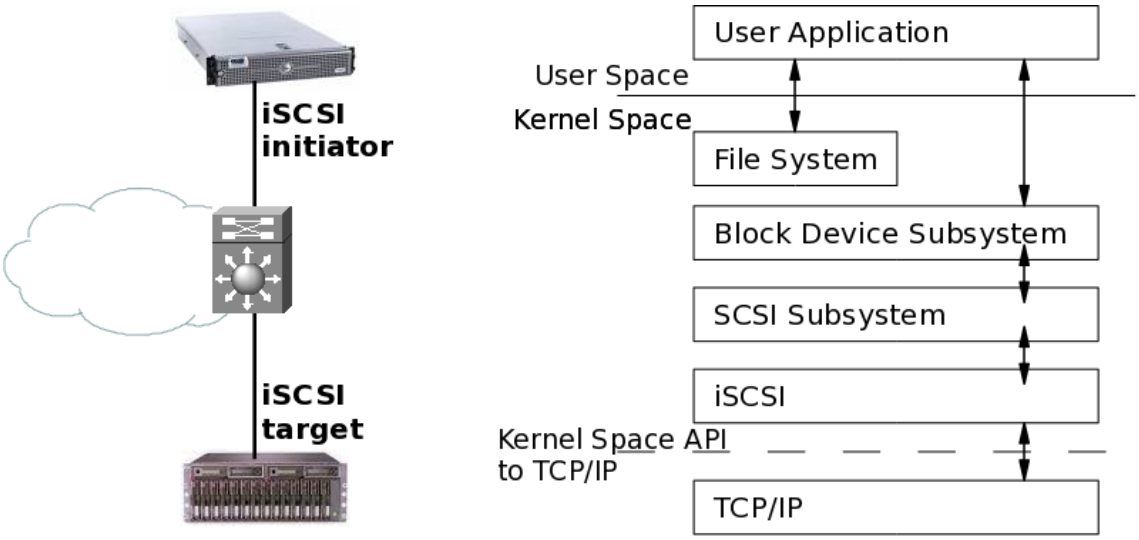
[1] http://www.docbook.org
[2] http://www.debian.org
[3] http://www.linux-france.org/prj/inetdoc/telechargement/admin.reseau.iscsi.pdf
[4] http://www.linux-france.org/prj/inetdoc/cours/archi.tp/
[5] http://www.linux-france.org/prj/inetdoc/cours/archi.tp/

4. Configurer les interfaces réseau de chaque poste : adresse, masque et passerelle par défaut. Valider la connectivité IP entre les deux postes puis avec les autres réseaux de l'infrastructure de travaux pratiques.

3. Technologie iSCSI et topologie de travaux pratiques

Cette section présente sommairement la technologie iSCSI et a pour but d'attribuer les rôles et les tâches de chacun des postes de travaux pratiques en fonction de la topologie mise en œuvre. Ce support de travaux pratiques fait suite à la présentation sur le *Stockage Réseau*[6] utilisée en cours.

3.1. Bases de la technologie iSCSI



Topologie iSCSI basique - vue complète[7]

La technologie iSCSI dont l'acronyme reprend la définition historique *Internet Small Computer System Interface* est un protocole réseau de stockage basé sur le modèle TCP/IP. Le principe de base consiste à encapsuler des commandes SCSI dans des paquets IP transmis entre un hôte et une unité de disque. Comme les paquets IP peuvent être perdus (et|ou) retransmis, ils peuvent très bien ne pas arriver dans l'ordre d'émission. Le protocole iSCSI doit donc conserver une trace de la séquence de transmission de commandes SCSI. Les commandes sont placées dans une file d'attente dans l'ordre d'émission.

La technologie iSCSI a initialement été développée par *IBM* et a ensuite été soumise à l'IETF (*Internet Engineering Task Force*). Le standard a été publié par le comité *IP Storage Working Group* en août 2002.

On peut identifier deux fonctions principales dans la technologie iSCSI. La première est la fonction *target*. C'est un système simple qui contient l'unité de disque. Ce système peut être matériel ou logiciel. Dans le cas de ces travaux pratiques, il s'agit d'un poste de travail utilisant son second disque dur comme unité de stockage SAN. La seconde fonction est baptisée *initiator*. Elle correspond au «client» qui utilise l'unité de stockage réseau. Ici, l'autre poste de travaux pratiques joue ce rôle de client.

Fondamentalement, iSCSI est un protocole de la famille *Storage Area Network* (SAN). Le client ou *initiator* accède à une unité de stockage en *mode bloc*. Ce mode de fonctionnement est quasi identique à la technologie *Fibre Channel*. Le type de réseau constitue la principale différence entre ces deux technologies. La technologie iSCSI s'appuie sur TCP/IP alors que *Fibre Channel* comprend une définition de réseau propre (FC) qui nécessite des équipements spécifiques.

Ces dernières années, la technologie iSCSI gagne en popularité relativement à son ainée pour plusieurs raisons.

- Le prix des configurations iSCSI peut être bien meilleur marché qu'avec la technologie *Fibre Channel*. Si les performances d'un réseau au Gigabit Ethernet suffisent, un réseau de stockage iSCSI devient très attractif.

  Attention cependant à bien identifier quand d'autres techniques sont associées à iSCSI pour accroître les débits réseau et donc les performances du stockage. Dans ces techniques complémentaires on trouve l'agrégation de

[6] http://www.linux-france.org/prj/inetdoc/articles/stockage/
[7] http://www.linux-france.org/prj/inetdoc/cours/admin.reseau.iscsi/images/topologie-iscsi.png

canaux baptisée *bonding*[8] sur les systèmes GNU/Linux et *etherchannel*[9] sur les équipements *Cisco*. Sous ces deux dénominations différentes se cache un même standard : l'IEEE 802.3ad *Dynamic link aggregation* qui rend l'utilisation de cette technique totalement transparente entre équipements hétérogènes. En appliquant ce principe d'agrégation de canaux, on peut pratiquement assimiler les performances de quatre liens Gigabit Ethernet à celles d'un lien *Fibre Channel*. Une autre technique consiste à utiliser aussi plusieurs liens dans une optique et redondance et de balance de charge. Elle est appelée *multipath*[10].

- L'utilisation d'une technologie réseau unique est nettement moins complexe à administrer. En effet, on optimise les coûts, les temps de formation et d'exploitation en utilisant une architecture de commutation homogène. C'est un des avantages majeurs de la technologie Ethernet sur ses concurrentes.
- Au début de son exploitation, le coût d'un réseau 10 Gigabit Ethernet est prohibitif relativement à toutes les autres solutions. On peut espérer que le coût d'acquisition des équipements 10GigE suivra le même profil que pour les débits antérieurs et que ces réseaux vont se démocratiser. Du point de vue hôte, le point déterminant est l'uniformisation de l'interface réseau. En effet, avec une interface 10GigE on ne devrait plus avoir de distinction entre NIC et HBA.

Aujourd'hui la technologie iSCSI est supportée par tous les systèmes d'exploitation communs. Côté GNU/Linux, plusieurs projets ont vu le jour dans les années qui ont suivi la publication du standard en 2002. Pour la partie *initiator* les développements des deux projets phares ont fusionné pour ne plus fournir qu'un seul code source ; celui disponible à l'adresse *Open-iSCSI*[11]. La partie *target* a suivi un processus analogue et le code source est disponible à l'adresse *iSCSI Enterprise Target*[12]. On peut simplement regretter que la partie *Kernelspace* n'ait pas encore été intégrée dans l'arborescence principale du noyau Linux. La mise en œuvre du rôle *target* nécessite donc des manipulations spécifiques pour compiler les modules si le système d'exploitation ne fournit pas le service DKMS (*Dynamic Kernel Module Support*).

On retrouve tous les éléments utiles sous forme de paquets dans la distribution Debian GNU/Linux.

```
$ aptitude search iscsi
p   iscsitarget          - iSCSI Enterprise Target userland tools
p   iscsitarget-dkms     - iSCSI Enterprise Target kernel module source - dkms version
p   iscsitarget-source   - iSCSI Enterprise Target kernel module source
p   open-iscsi           - High performance, transport independent iSCSI implementation
```

3.2. Infrastructure de stockage étudiée

Le séquencement des opérations à réaliser lors de la séance de travaux pratiques est décrit dans le tableau ci-dessous. Les deux postes occupent chacun un rôle distinct. Comme le rôle *initiator* demande moins de travail de préparation, c'est à ce poste que l'on confie les essais des outils de *micro-benchmark*.

Tableau 2. Attribution des rôles

Rôle *initiator*	**Rôle *target***
Préparation d'une unité de stockage locale pour évaluer les différences entre les accès DAS et SAN	Préparation d'une unité de stockage iSCSI
Installation des outils	
Mise au point des scripts d'évaluation des performances du stockage	Compilation et configuration des outils de déploiement d'une unité de stockage iSCSI
Validation de la configuration SAN iSCSI	
Étude comparative des performances	

4. Préparation d'une unité de stockage

Dans cette section on présente les manipulations à effectuer pour préparer une unité de stockage à son utilisation dans une configuration DAS (et|ou) SAN.

[8] http://www.linuxfoundation.org/collaborate/workgroups/networking/bonding
[9] http://www.cisco.com/en/US/tech/tk389/tk213/technologies_white_paper09186a0080092944.shtml
[10] http://sources.redhat.com/lvm2/wiki/MultipathUsageGuide
[11] http://www.open-iscsi.org/
[12] http://iscsitarget.sourceforge.net/

Avertissement

Les commandes données dans les réponses correspondent à l'utilisation de machines virtuelles. Les unités de disques apparaissent donc sous le nom `/dev/vd[a-z]`. Les unités de disques physiques d'un système réel apparaissent sous le nom `/dev/sd[a-z]`.

4.1. Destruction de la table des partitions

Sachant que les disques des postes de travaux pratiques sont utilisés régulièrement, il est préférable de se placer dans le contexte d'utilisation d'une unité de disque vierge de tout système de fichiers.

1. Quelle est la syntaxe d'appel de l'outil parted qui permet de visualiser la table de partition d'une unité de disque ?

```
# parted /dev/vda print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 34,4GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  256MB   255MB   primary   ext2         boot
 2      257MB   34,4GB  34,1GB  extended
 5      257MB   34,4GB  34,1GB  logical               lvm
```

2. Quelle est la syntaxe de la commande **dd** qui permet d'effacer complètement la table des partitions d'une unité de disque ?

La commande suivante écrit des 0 dans les 4 premiers blocs de 512 octets de l'unité de disque.

```
# dd if=/dev/zero of=/dev/vdb bs=512 count=4
4+0 enregistrements lus
4+0 enregistrements écrits
2048 octets (2,0 kB) copiés, 0,0617608 s, 33,2 kB/s

# parted /dev/vdb print
Error: /dev/vdb: unrecognised disk label
```

4.2. Création de la table des partitions et formatage

Une fois que l'on dispose d'une unité de disque vierge, on peut passer à l'étape de création de la table des partitions. Dans le contexte de ces travaux pratiques, cette opération doit être effectuée deux fois sur les postes de travail pour les deux types d'unité de stockage utilisées.

1. Le second disque physique des postes de travail est destiné à intégrer l'unité logique RAID1.
2. Le disque réseau iSCSI est disponible une fois que la configuration du rôle *initiator* est active.

Cette manipulation est l'opération de plus bas niveau qui caractérise un accès réseau au stockage en *mode bloc* et non en mode fichier.

1. Quelles sont les instructions de l'outil parted qui permettent de créer une partition primaire unique couvrant la totalité de l'espace de stockage de l'unité de disque ?

```
# parted /dev/vdb
GNU Parted 2.3
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel gpt
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 34,4GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start  End  Size  File system  Name  Flags
```

```
(parted) mkpart primary ext4 1 -1
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 34,4GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  34,4GB  34,4GB               primary

(parted) quit
Information: You may need to update /etc/fstab.
```

2. Quelle est la syntaxe de la commande de formatage de la partition créée lors de l'étape précédente ?

```
# mkfs.ext4 /dev/vdb1
mke2fs 1.41.12 (17-May-2010)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
Taille de fragment=4096 (log=2)
« Stride » = 0 blocs, « Stripe width » = 0 blocs
2097152 i-noeuds, 8388096 blocs
419404 blocs (5.00%) réservés pour le super utilisateur
Premier bloc de données=0
Nombre maximum de blocs du système de fichiers=4294967296
256 groupes de blocs
32768 blocs par groupe, 32768 fragments par groupe
8192 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624

Écriture des tables d'i-noeuds : complété
Création du journal (32768 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété

Le système de fichiers sera automatiquement vérifié tous les 36 montages ou
après 180 jours, selon la première éventualité. Utiliser tune2fs -c ou -i
pour écraser la valeur.
```

3. Quelle est la syntaxe de la commande de visualisation des attributs du système de fichiers créé lors du formatage ?

Les informations utiles sur les attributs sont fournies à la page *Ext4 Howto*[13].

```
# tune2fs -l /dev/vdb1
tune2fs 1.41.12 (17-May-2010)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          760d652d-f64e-4bb0-a017-7d981f0da4ec
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype \
     extent flex_bg sparse_super large_file huge_file \
     uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              2097152
Block count:              8388096
Reserved block count:     419404
Free blocks:              8211438
Free inodes:              2097141
```

[13] https://ext4.wiki.kernel.org/index.php/Ext4_Howto

```
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      1022
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Wed Apr  6 13:50:31 2011
Last mount time:          n/a
Last write time:          Wed Apr  6 13:50:53 2011
Mount count:              0
Maximum mount count:      36
Last checked:             Wed Apr  6 13:50:31 2011
Check interval:           15552000 (6 months)
Next check after:         Mon Oct  3 13:50:31 2011
Lifetime writes:          644 MB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     28
Desired extra isize:      28
Journal inode:            8
Default directory hash:   half_md4
Directory Hash Seed:      e7374b09-c08e-461b-9a43-feaaf8ce0f1d
Journal backup:           inode blocks
```

5. Configuration du système initiator

Dans cette partie, on prépare le système auquel on a attribué le rôle *initiator*.

5.1. Sélection du paquet et lancement du service

1. Comment identifier et installer le paquet correspondant au rôle *initiator* ?

En effectuant une recherche simple dans le catalogue des paquets disponibles, on obtient la liste des paquets dont le nom contient la chaîne de caractères `iscsi`.

```
# aptitude search iscsi
p   iscsitarget         - iSCSI Enterprise Target userland tools
p   iscsitarget-dkms    - iSCSI Enterprise Target kernel module source - dkms version
p   iscsitarget-source  - iSCSI Enterprise Target kernel module source
p   open-iscsi          - High performance, transport independent iSCSI implementation
```

On remarque que le paquet `open-iscsi` est le seul qui ne soit pas identifié comme appartenant à la catégorie *target*.

```
# aptitude install open-iscsi
```

2. Comment lancer le service *initiator* et valider son fonctionnement ?

À partir de la liste des fichiers du paquet on peut identifier les éléments de démarrage et de configuration du service.

```
# # dpkg -L open-iscsi
/.
/etc
/etc/init.d
/etc/init.d/open-iscsi
/etc/init.d/umountiscsi.sh
/etc/network
/etc/network/if-up.d
/etc/default
/etc/default/open-iscsi
/etc/iscsi
/etc/iscsi/iscsid.conf
```

```
/etc/iscsi/initiatorname.iscsi
/usr
/usr/share
/usr/share/man
/usr/share/man/man8
/usr/share/man/man8/iscsiadm.8.gz
/usr/share/man/man8/iscsid.8.gz
/usr/share/man/man8/iscsi_discovery.8.gz
/usr/share/man/man8/iscsistart.8.gz
/usr/share/man/man8/iscsi-iname.8.gz
/usr/share/initramfs-tools
/usr/share/initramfs-tools/hooks
/usr/share/initramfs-tools/hooks/iscsi
/usr/share/initramfs-tools/scripts
/usr/share/initramfs-tools/scripts/local-top
/usr/share/initramfs-tools/scripts/local-top/iscsi
/usr/share/doc
/usr/share/doc/open-iscsi
/usr/share/doc/open-iscsi/copyright
/usr/share/doc/open-iscsi/README.Debian.gz
/usr/share/doc/open-iscsi/README.gz
/usr/share/doc/open-iscsi/changelog.Debian.gz
/usr/share/doc/open-iscsi/changelog.gz
/usr/sbin
/usr/sbin/iscsi-iname
/usr/sbin/iscsid
/usr/sbin/iscsi_discovery
/usr/sbin/iscsistart
/usr/bin
/usr/bin/iscsiadm
/var
/var/lib
/var/lib/open-iscsi
```

Le lancement du service se fait de façon classique à partir de l'arbrescence des scripts des niveaux de démarrage (*runlevels*).

```
# /etc/init.d/open-iscsi restart
```

On peut ensuite consulter les journaux système pour valider l'initialisation du ou des démons.

```
# grep -i iscsi /var/log/syslog
2ndInitiator kernel: [50475.436562] Loading iSCSI transport class v2.0-870.
2ndInitiator kernel: [50475.471559] iscsi: registered transport (tcp)
2ndInitiator kernel: [50475.612327] iscsi: registered transport (iser)
2ndInitiator iscsid: iSCSI logger with pid=6725 started!
2ndInitiator iscsid: transport class version 2.0-870. iscsid version 2.0-871
2ndInitiator iscsid: iSCSI daemon with pid=6726 started!
```

On retrouve les informations correspondantes aux messages ci-dessus dans la liste des processus actifs.

```
# ps aux | grep -i iscsi
root      6717  0.0  0.0      0     0 ?        S    01:47   0:00 [iscsi_eh]
root      6725  0.0  0.1   4056   512 ?        Ss   01:47   0:00 /usr/sbin/iscsid
root      6726  0.0  0.4   4548  2500 ?        S<Ls 01:47   0:00 /usr/sbin/iscsid
root      6750  0.0  0.1   9612   896 pts/0    S+   01:48   0:00 grep -i iscsi
```

5.2. Tests de fonctionnement du service

1. Quelle est la commande principale du rôle *initiator* qui permet de tester la connectivité iSCSCI ?

En consultant la liste des fichiers du paquet `open-iscsi` donnée ci-dessus, on ne relève qu'un seul outil binaire : la commande **iscsiadm**.

2. Quelles sont les options de découverte proposées avec cette commande de configuration ? Donner un exemple fournissant l'identifiant de l'unité de stockage réseau accessible.

```
# iscsiadm -m discovery --type sendtargets --portal=192.200.0.12:3260
192.200.0.12:3260,1 iqn.2011-04.lab.stri:1stInitiator.disk
```

```
192.200.0.12:3260,1 iqn.2011-04.lab.stri:2ndInitiator.disk
```

3. Quelles sont les options de connexion proposées avec cette même commande ? Donner un exemple illustrant l'établissement d'une connexion.

```
# iscsiadm -m node -T iqn.2011-04.lab.stri:1stInitiator.disk -p 192.200.0.12 -l
Logging in to [iface: default, target: iqn.2011-04.lab.stri:1stInitiator.disk, portal: 192.200.0.12,3260]
Login to [iface: default, target: iqn.2011-04.lab.stri:1stInitiator.disk, portal: 192.200.0.12,3260]: succes
```

4. Comment obtenir les caractéristiques de l'unité de stockage iSCSI utilisée ?

On peut consulter les journaux système. Voici un extrait du fichier `/var/log/syslog`.

```
# egrep '(sd|scsi)' /var/log/syslog
1stInitiator kernel: [51051.744035] iscsi: registered transport (tcp)
1stInitiator kernel: [51051.783694] iscsi: registered transport (iser)
1stInitiator iscsid: iSCSI logger with pid=6546 started!
1stInitiator iscsid: transport class version 2.0-870. iscsid version 2.0-871
1stInitiator iscsid: iSCSI daemon with pid=6547 started!
1stInitiator kernel: [51066.871258] scsi3 : iSCSI Initiator over TCP/IP
1stInitiator kernel: [51067.138424] scsi 3:0:0:0: Direct-Access     IET      VIRTUAL-DISK     0    PQ: 0 ANS
1stInitiator kernel: [51067.139571] sd 3:0:0:0: Attached scsi generic sg1 type 0
1stInitiator kernel: [51067.144997] sd 3:0:0:0: [sda] 67108864 512-byte logical blocks: (34.3 GB/32.0 GiB)
1stInitiator kernel: [51067.146307] sd 3:0:0:0: [sda] Write Protect is off
1stInitiator kernel: [51067.146310] sd 3:0:0:0: [sda] Mode Sense: 77 00 00 08
1stInitiator kernel: [51067.146965] sd 3:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't su
1stInitiator kernel: [51067.148615]  sda: unknown partition table
1stInitiator kernel: [51067.152320] sd 3:0:0:0: [sda] Attached SCSI disk
1stInitiator iscsid: connection1:0 is operational now
```

5.3. Configuration système

Une fois la connexion à la ressource iSCSI testée, on peut passer à la configuration système de façon à retrouver le volume de stockage après une réinitialisation du système *initiator*.

1. Comment rendre la connexion à l'unité de stockage automatique lors de l'initialisation du système *initiator* ?

Le fichier `/etc/iscsi/iscsid.conf` contient une directive dans la section *Startup settings* qui rend automatique l'accès à une ressource déjà enregistrée. Voici le contenu de cette section extraite du fichier de configuration.

```
#*****************
# Startup settings
#*****************

# To request that the iscsi initd scripts startup a session set to "automatic".
node.startup = automatic
```

2. Comment authentifier la connexion entre les systèmes *initiator* et *target* ?

Le mécanisme d'authentification le plus communément utilisé dans le déploiement des connexions iSCSI s'appuie sur CHAP (*Challenge-Handshake Authentication Protocol*. Il s'agit d'une méthode d'authentification entre deux hôtes pairs sans échange de mot de passe en clair sur le réseau. Cette méthode suppose que les deux hôtes utilisent le même mot de passe.

Le nom d'utilisateur et le mot de passe sont respectivement définis dans les fichiers `/etc/ietd.conf` et `/etc/iscsi/iscsid.conf` des systèmes *target* et *initiator*.

- Système *target* :

```

Target iqn.2010-04.lab.stri:san.disk
        # Users, who can access this target. The same rules as for discovery
        # users apply here.
        # Leave them alone if you don't want to use authentication.
        IncomingUser etu-san MyS3cr3t-54n
<snipped/>
```

- Système *initiator* :

```
# *************
# CHAP Settings
# *************

# To enable CHAP authentication set node.session.auth.authmethod
# to CHAP. The default is None.
node.session.auth.authmethod = CHAP

# To set a CHAP username and password for initiator
# authentication by the target(s), uncomment the following lines:
node.session.auth.username = etu-san
node.session.auth.password = MyS3cr3t-54n
```

Le même principe peut être appliqué au mécanisme de découverte en appliquant un couple *login/password* identique ou non à la suite de ce fichier de configuration.

6. Configuration du système target

Dans cette partie, on prépare le système auquel on a attribué le rôle *target*. Cette préparation comprend deux étapes : la construction et l'installation d'un paquet contenant le module de la partie *KernelSpace* du service et la configuration des outils de la partie *UserSpace* pour lancer le démon `ietd`.

6.1. Sélection des paquets et compilation du module

1. Quels sont les paquets de la distribution Debian GNU/Linux qui permettent de configurer un système avec le rôle *target* ?

En effectuant une recherche simple dans le catalogue des paquets disponibles, on obtient la liste des paquets dont le nom contient la chaîne de caractères `iscsi`.

```
$ aptitude search iscsi
p   iscsitarget         - iSCSI Enterprise Target userland tools
p   iscsitarget-dkms    - iSCSI Enterprise Target kernel module source - dkms version
p   iscsitarget-source  - iSCSI Enterprise Target kernel module source
p   open-iscsi          - High performance, transport independent iSCSI implementation
```

Dans la liste ci-dessus, on distingue deux éléments. Le premier paquet : `iscsitarget-dkms` contient le code source du module qui doit être chargé en mémoire pour faire fonctionner le service. Ce module doit être compilé en fonction de la version du noyau actif sur le système. La version DKMS associée à la gestion des dépendances permet de construire automatiquement un paquet correspodant à la version du noyau Linux utilisée. Le second paquet : `iscsitarget` contient le logiciel de gestion du service.

2. Quelle est la commande d'installation des paquets nécessaires à la compilation des modules et à la configuration du rôle *target* ?

```
# aptitude install iscsitarget iscsitarget-dkms
Les NOUVEAUX paquets suivants vont être installés :
  binutils{a} cpp{a} cpp-4.3{a} cpp-4.5{a} dkms{a} fakeroot{a} gcc{a}
  gcc-4.3{a} gcc-4.3-base{a} gcc-4.5{a} iscsitarget iscsitarget-dkms
  libc-dev-bin{a} libc6-dev{a} libcloog-ppl0{a} libelfg0{a} libgmp10{a}
  libgmp3c2{a} libgmpxx4ldbl{a} libgomp1{a} libmpc2{a} libmpfr4{a} libppl-c2{a}
  libppl7{a} linux-headers-2.6-amd64{a} linux-headers-2.6.32-5-amd64{a}
  linux-headers-2.6.32-5-common{a} linux-kbuild-2.6.32{a} linux-libc-dev{a}
  make{a} manpages-dev{a} menu{a}
0 paquets mis à jour, 32 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de télécharger 35,7 Mo d'archives. Après dépaquetage, 106 Mo seront utilisés.
Voulez-vous continuer ? [Y/n/?]
<snipped/>
```

On retrouve le module compilé à l'aide du service DKMS dans l'arborescence usuelle des modules du noyau Linux.

```
# ll /lib/modules/`uname -r`/updates/dkms
total 136K
-rw-r--r-- 1 root root 131K  5 avril 16:21 iscsi_trgt.ko
```

6.2. Configuration du service

1. Quel est le paquet qui contient les éléments de configuration du service dans l'espace utilisateur ?

On repart de la même liste des paquets iSCSI installés pour identifier celui qui contient le démon du service.

```
# aptitude search ~iiscsi
i   iscsitarget        - iSCSI Enterprise Target userland tools
i   iscsitarget-dkms   - iSCSI Enterprise Target kernel module source - dkms version
```

Une fois le paquet identifié, on peut lister son contenu.

```
# dpkg -L iscsitarget
/.
/etc
/etc/init.d
/etc/init.d/iscsitarget
/etc/default
/etc/default/iscsitarget
/etc/sysctl.d
/etc/sysctl.d/30-iscsitarget.conf
/etc/iet
/etc/iet/ietd.conf
/etc/iet/targets.allow
/etc/iet/initiators.allow
/usr
/usr/share
/usr/share/man
/usr/share/man/man5
/usr/share/man/man5/ietd.conf.5.gz
/usr/share/man/man8
/usr/share/man/man8/ietd.8.gz
/usr/share/man/man8/ietadm.8.gz
/usr/share/lintian
/usr/share/lintian/overrides
/usr/share/lintian/overrides/iscsitarget
/usr/share/doc
/usr/share/doc/iscsitarget
/usr/share/doc/iscsitarget/ChangeLog.gz
/usr/share/doc/iscsitarget/copyright
/usr/share/doc/iscsitarget/README.Debian
/usr/share/doc/iscsitarget/README.vmware
/usr/share/doc/iscsitarget/README.gz
/usr/share/doc/iscsitarget/changelog.Debian.gz
/usr/share/doc/iscsitarget/changelog.gz
/usr/share/doc/iscsitarget/NEWS.Debian.gz
/usr/sbin
/usr/sbin/ietadm
/usr/sbin/ietd
```

2. Quelle est l'opération à effectuer pour lancer automatiquement le service lors de l'initialisation du système ?

Le répertoire `/etc/default` contient une liste de fichiers de contrôle qui servent à paramétrer les services lors de leur démarrage. Dans la liste ci-dessus, on relève le fichier `/etc/default/iscsitarget` dont le contenu est à éditer pour activer le service.

```
# cat /etc/default/iscsitarget
ISCSITARGET_ENABLE=true
```

3. Quel est le fichier de configuration principal du service ? Quelles sont les entrées utiles au fonctionnement du rôle *target* ?

Toujours à partir de la liste des fichiers du paquets, on recherche les fichiers appartenant au répertoire `/etc/` pour identifier celui qui sert à configurer le démon.

Ici, on repère facilement le fichier `/etc/iet/ietd.conf` que l'on doit éditer pour donner accès à l'unité de stockage réseau.

Parmi les multiples choix disponibles pour la désignation des unités de stockage que l'on souhaite rendre accessible via le réseau, on peut retenir deux solutions classiques. Pour ces deux solutions on donne un exemple de fichier de configuration.

- On peut désigner une unité physique locale en mode bloc (DAS) comme ressource cible iSCSI.

```
# egrep -v '(^*#|^$)' /etc/iet/ietd.conf
Target iqn.2011-04.lab.stri:san.disk
        Lun 0 Path=/dev/vdb,Type=blockio
 Alias san.disk
```

- On peut désigner un fichier local comme ressource cible iSCSI. Cette seconde option est intéressante pour fournir plusieurs unités logiques (LUN) à partir d'un même volume local (DAS).

```
# egrep -v '(^*#|^$)' /etc/iet/ietd.conf
Target iqn.2011-04.lab.stri:1stInitiator.disk
        Lun 0 Path=/var/lib/target/1stInitiator.disk,Type=fileio
        Alias 1stInitiator.disk
Target iqn.2011-04.lab.stri:2ndInitiator.disk
        Lun 1 Path=/var/lib/target/2ndInitiator.disk,Type=fileio
        Alias 2ndInitiator.disk
```

Deux remarques sur les choix de configuration :

- La technologie iSCSI dispose d'un schéma de nommage propre défini dans le document standard RFC3721[14]. Le format retenu ici est baptisé iqn (*iSCSI Qualified Name*). Il s'agit d'une chaîne qui débute par "iqn." suivie d'une date au format AAAA-MM, du nom de l'autorité qui a attribué le nom (le nom de domaine à l'envers), puis une autre chaîne unique qui identifie le nœud de stockage.
- On a choisi de n'utiliser aucun mécanisme d'authentification sachant que la configuration se fait dans un contexte de travaux pratiques et non d'exploitation sur un réseau réel.

4. Comment lancer le service et s'assurer de son bon fonctionnement ?

Pour ce qui est du lancement du service, on retrouve l'appel classique au script appartenant aux niveaux de démarrage (*runlevels*).

```
# /etc/init.d/iscsitarget restart
```

Ensuite, pour le contrôle de l'état du service, on passe par les étapes classiques.

- Consultation des journaux système.

```
# grep -i iscsi /var/log/syslog
Apr  6 11:46:32 targetSrv kernel: [   40.416451] iSCSI Enterprise Target Software - version 1.4.20.2
Apr  6 11:46:32 targetSrv kernel: [   40.416691] iscsi_trgt: Registered io type fileio
Apr  6 11:46:32 targetSrv kernel: [   40.416693] iscsi_trgt: Registered io type blockio
Apr  6 11:46:32 targetSrv kernel: [   40.416695] iscsi_trgt: Registered io type nullio
```

- Recherche dans la liste des processus actifs.

```
# ps aux | grep ietd
root      1022  0.0  0.1   3916   544 ?        Ss   Apr06   0:00 /usr/sbin/ietd
root      6325  0.0  0.1   9616   884 pts/0    S+   00:43   0:00 grep ietd
```

- Comme il s'agit d'un service réseau, on peut identifier les ports ouverts correspondant au démon.

```
# lsof -i | grep ietd
ietd    1022        root    7u  IPv4   3875      0t0  TCP *:3260 (LISTEN)
ietd    1022        root    8u  IPv6   3876      0t0  TCP *:3260 (LISTEN)
```

5. Comment autoriser l'accès à la ressource de stockage pour tout le réseau local ?

On utilise le fichier `/etc/initiators.allow` pour configurer les accès réseau. Une fois de plus, on va au plus simple en retenant l'adresse du réseau local sachant que ce réseau ne contient que deux hôtes.

```
# egrep -v ^# /etc/iet/initiators.allow
ALL 192.200.0.0/27
```

[14] http://www.ietf.org/rfc/rfc3721.txt

6. Comment visualiser l'état de la session d'accès à la ressource de stockage iSCSI ?

On consulte le fichier `/proc/net/iet/session` qui donne l'état courant de la session avec l'adresse IP du système initiator.

```
# cat /proc/net/iet/session
tid:2 name:iqn.2011-04.lab.stri:2ndInitiator.disk
tid:1 name:iqn.2011-04.lab.stri:1stInitiator.disk
```

7. Configuration d'une unité logique RAID1

Dans cette partie, on crée une unité logique RAID1 composée d'une unité de disque locale et d'une unité de disque iSCSI dans le but d'illustrer une solution de réplication synchrone. En effet, dans un volume RAID1 chaque disque contient à tout moment exactement les mêmes données. Ici, le contenu de l'unité de disque locale est identique à celui de l'unité de disque réseau. La réplication ainsi réalisée est dite synchrone puisque toute écriture locale est dupliquée sur le réseau de stockage iSCSI.

7.1. Sélection du paquet et création de l'unité de stockage

1. Quel est le paquet qui contient les outils de configuration et de gestion des différents types d'unités RAID logicielles ?

On utilise à nouveau les recherches dans les listes de paquets de la distribution.

```
# apt-cache search software raid
<snipped/>
mdadm - tool to administer Linux MD arrays (software RAID)
<snipped/>
```

Une fois le paquet identifié et installé, on peut lister son contenu et isoler les commandes utilisateur.

```
# dpkg -L mdadm | grep bin
/sbin
/sbin/mdmon
/sbin/mdadm-startall
/sbin/mdadm
```

2. Quelle est la syntaxe de la commande **mdadm** qui permet de créer l'unité logique RAID1 ?

```
# mdadm --create /dev/md0 --level=raid1 --raid-devices=2 /dev/sda /dev/vdb
mdadm: partition table exists on /dev/sda but will be lost or
       meaningless after creating array
mdadm: Note: this array has metadata at the start and
    may not be suitable as a boot device.  If you plan to
    store '/boot' on this device please ensure that
    your boot-loader understands md/v1.x metadata, or use
    --metadata=0.90
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

7.2. Manipulations sur l'unité de stockage RAID1

1. Comment connaître l'état de l'unité logique RAID1 ?

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

2. Comment afficher la liste des propriétés de l'unité logique RAID1 ?

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

3. Comment rendre la configuration système permanente ?

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

4. Comment rendre la configuration système permanente ?

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

5. Comment arrêter le gestionnaire RAID ?

```
# mdadm --stop /dev/md0
```

6. Comment retirer une unité du gestionnaire RAID ?

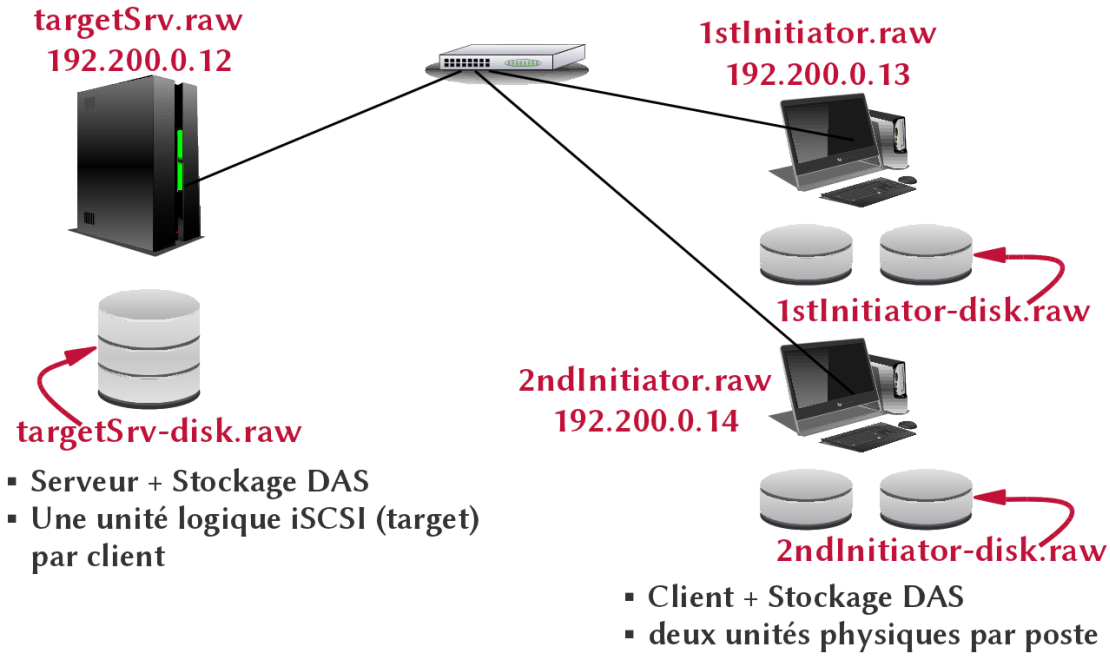
```
# mdadm /dev/md0 --fail /dev/sda --remove /dev/sda
```

8. Manipulations sur machines virtuelles

Il est possible de réaliser l'ensemble des manipulations de ce support à l'aide de trois instances de machines virtuelles et du commutateur virtuel *Virtual Distributed Ethernet* présenté dans l'article *Virtualisation système et enseignements pratiques*[15]

8.1. Préparation du système hôte

L'infrastructure à implanter sur le système hôte est la suivante.



Topologie virtualisation iSCSI - vue complète[16]

On débute avec la création des fichiers image des trois systèmes virtuels. Les fichiers de type `.qcow2` sont des images compressées et sont plus faciles à transférer que les fichiers de type `.raw`.

```
mkdir -p ~/vm/iscsi
cd ~/vm/iscsi

cp ../vm0-debian-testing-amd64-base.qcow2 targetSrv.qcow2
cp ../vm0-debian-testing-amd64-base.qcow2 1stInitiator.qcow2
cp ../vm0-debian-testing-amd64-base.qcow2 2ndInitiator.qcow2

for file in *.qcow2; do qemu-img convert -f qcow2 -O raw $file `basename $file .qcow2`.raw; done

rm *.qcow2
```

On crée ensuite les fichiers correspondant aux unités de stockage supplémentaires.

```
qemu-img create -f raw targetSrv-disk.raw 80G
qemu-img create -f raw 1stInitiator-disk.raw 32G
```

[15] http://www.linux-france.org/prj/inetdoc/articles/vm/
[16] http://www.linux-france.org/prj/inetdoc/cours/admin.reseau.iscsi/images/topologie-iscsi-vm.png

```
qemu-img create -f raw 2ndInitiator-disk.raw 32G
```

Enfin, il ne reste qu'à mettre en place le script de lancement de ces trois systèmes avec leurs unités de stockages respectives.

```
#!/bin/bash

../scripts/startup.sh targetSrv.raw 512 2 \
    -drive file=targetSrv-disk.raw,if=virtio,boot=off

../scripts/startup.sh 1stInitiator.raw 512 3 \
    -drive file=1stInitiator-disk.raw,if=virtio,boot=off

../scripts/startup.sh 2ndInitiator.raw 512 4 \
    -drive file=2ndInitiator-disk.raw,if=virtio,boot=off
```

Ce script fait lui-même appel au script commun `startup.sh` qui sert à initialiser une instance de machine virtuelle en utilisant comme paramètres le nom du fichier image, la quantité de RAM et le port du commutateur VDE. Le code du script `startup.sh` de lancement d'une machine virtuelle[17] est donné en annexe de l'article *Virtualisation système et enseignements pratiques*[18].

8.2. Serveur target iSCSI

Cette instance de système virtuel joue le rôle du système *target* iSCSI. On reprend donc ici les manipulations présentées dans la Section 6, « Configuration du système target » après avoir préparé le volume de stockage et les fichiers ressource cible iSCSI.

Lors de l'initialisation des trois systèmes virtuels, on a ajouté un fichier disque de 80Go à l'instance baptisée `targetSrv`. C'est ce volume de stockage qui est utilisé pour ouvrir l'accès aux deux unités ressources cible iSCSI.

On reprend les manipulations de la Section 4, « Préparation d'une unité de stockage » pour le partitionnement et le formatage du volume physique virtuel repéré `/dev/vdb`.

```
dd if=/dev/zero of=/dev/vdb bs=512 count=4

parted /dev/vdb mklabel gpt
parted /dev/vdb mkpart primary ext4 1 -1

mkfs.ext4 /dev/vdb1

mkdir /var/lib/target

mount /dev/vdb1 /var/lib/target
```

Pour rendre ce montage «permanent», on édite le fichier `/etc/fstab` après avoir identifié le volume de stockage.

```
blkid | grep vdb
/dev/vdb1: UUID="10f56553-ca7c-4827-b43f-8ef664fcef6f" TYPE="ext4"

echo UUID="10f56553-ca7c-4827-b43f-8ef664fcef6f" \
  /var/lib/target/ \
  ext4 \
  defaults \
  0      2 >>/etc/fstab
```

Une fois le montage du volume effectué à partir du répertoire `/var/lib/target`, on peut créer les fichiers ressources cible iSCSI à l'aide de la commande **dd**. On crée des fichier de type *Sparse file*[19] pour optimiser l'occupation disque réelle sur le système hôte.

Pour respecter le scénario des travaux pratiques, on utilise des fichiers de 32Go composés de blocs de 4Ko qui correspondent au volumes de stockage des deux autres instances de machines virtuelles. La «formule de calcul» devient : `(32 x 1024 x 1024 x 1024) / (4 x 1024) = 8388608`.

[17] http://www.linux-france.org/prj/inetdoc/articles/vm/vm.appendix.html#vm.appendix.startup.sh
[18] http://www.linux-france.org/prj/inetdoc/articles/vm/
[19] http://en.wikipedia.org/wiki/Sparse_file

```
cd /var/lib/target/

dd if=/dev/null of=1stInitiator.disk bs=4k seek=8388608

dd if=/dev/null of=2ndInitiator.disk bs=4k seek=8388608

ll *.disk
-rw-r--r-- 1 root root 32G  7 avril 01:28 1stInitiator.disk
-rw-r--r-- 1 root root 32G  7 avril 01:28 2ndInitiator.disk

du -hs *.disk
0       1stInitiator.disk
0       2ndInitiator.disk
```

Comme indiqué dans la Section 6, « Configuration du système target », on installe les paquets `iscsitarget` et `iscsitarget-dkms`. Le fichier de configuration du démon `ietd` contient les références aux deux fichiers ci-dessus.

```
egrep -v '(^*#|^$)' /etc/iet/ietd.conf
Target iqn.2011-04.lab.stri:1stInitiator.disk
        Lun 0 Path=/var/lib/target/1stInitiator.disk,Type=fileio
        Alias 1stInitiator.disk
Target iqn.2011-04.lab.stri:2ndInitiator.disk
        Lun 1 Path=/var/lib/target/2ndInitiator.disk,Type=fileio
        Alias 2ndInitiator.disk
```

Une fois le service démarré, ces deux ressources cibles iSCSI sont utilisables.

```
cat /proc/net/iet/session
tid:2 name:iqn.2011-04.lab.stri:2ndInitiator.disk
tid:1 name:iqn.2011-04.lab.stri:1stInitiator.disk
```

8.3. Clients initiator iSCSI

Ces deux instances de système virtuel jouent le rôle du système *initiator* iSCSI. On reprend donc ici les manipulations présentées dans la Section 5, « Configuration du système initiator » avant de passer à la configuration de l'unité logique RAID1.

9. Documents de référence

Architecture réseau des travaux pratiques

Architecture réseau des travaux pratiques[20] : présentation de l'implantation des équipements d'interconnexion réseau dans l'armoire de brassage et du plan d'adressage IP prédéfini pour l'ensemble des séances de travaux pratiques. Ce document est utilisé dans la Section 2, « Adressage IP des postes de travail ».

Configuration d'une interface réseau

Configuration d'une interface de réseau local[21] : tout sur la configuration des interfaces réseau de réseau local ; notamment les explications sur les opérations «rituelles» de début de travaux pratiques :

```
# ifdown eth0
# ifconfig eth0 192.168.0.2 netmask 255.255.255.240
# route add default gw 192.168.0.1
# ping 192.168.0.1
# ping 172.16.80.1
# ping www.cict.fr
```

Comme dans le cas précédent, ce document est utile pour effectuer les opérations demandées dans la Section 2, « Adressage IP des postes de travail ».

Introduction to iSCSI

L'article intitulé *Introduction to iSCSI*[22] du site Linux Magazine présente les points clés de la technologie iSCSI. Il complète la Section 3, « Technologie iSCSI et topologie de travaux pratiques ».

[20] http://www.linux-france.org/prj/inetdoc/cours/archi.tp/
[21] http://www.linux-france.org/prj/inetdoc/cours/config.interface.lan/
[22] http://www.linux-mag.com/id/7605/

Le support du protocole iSCSI dans Linux
: L'article *Le support du protocole iSCSI dans Linux*[23] de la rubrique «Administration Réseau» du site *Unix Garden* présente l'ensemble des manipulations effectuées dans ce support de travaux pratiques. C'est une référence indispensable.

iSCSI - Debian Wiki
: La page *iSCSI and Debian*[24] contient deux sous-rubriques sur les rôles *initiator* et *target*. Ces sous-rubriques correspondent respectivement à la Section 5, « Configuration du système initiator » et à la Section 6, « Configuration du système target ». On y retrouve les paquets à utiliser et les principales commandes de configuration.

[patch] 2.6.32 removed sync_page_range
: La correction du fichier source `file-io.c`[25] permet de compiler le module du noyau Linux nécessaire au rôle *target*. Un *patch* plus succinct est proposé dans la Section 6.1, « Sélection des paquets et compilation du module ».

A. Configuration du système target sans service DKMS

Dans cette annexe, on prépare un système auquel on a attribué le rôle *target*. Les opérations présentées ici ne font pas appel au service DKMS (*Dynamic Kernel Module Support*) qui doit être disponible avec les distributions GNU/Linux récentes. Le contenu de cette annexe devrait donc devenir *obsolète*.

La préparation comprend deux étapes : la construction et l'installation d'un paquet contenant le module de la partie *KernelSpace* du service et la configuration des outils de la partie *UserSpace* pour lancer le démon `ietd`.

1. Quels sont les paquets de la distribution Debian GNU/Linux qui permettent de configurer un système avec le rôle *target* ?

En effectuant une recherche simple dans le catalogue des paquets disponibles, on obtient la liste des paquets dont le nom contient la chaîne de caractères `iscsi`.

```
$ aptitude search iscsi
p   iscsitarget         - iSCSI Enterprise Target userland tools
p   iscsitarget-source  - iSCSI Enterprise Target kernel module source
p   open-iscsi          - High performance, transport independent iSCSI implementation
```

On remarque que le paquet `iscsitarget-source` contient le code source du module qui doit être chargé en mémoire pour faire fonctionner le service. Ce module doit être compilé en fonction de la version du noyau actif sur le système. Il faut donc installer de tous les outils de la chaîne de développement nécessaires. Heureusement, la distribution Debian fournit un outil générique dédié à la construction de paquets de modules : *Module-Assistant*[1].

```
$ aptitude search module-assistant
i   module-assistant    - tool to make module package creation easier
```

2. Quelle est la commande d'installation des paquets nécessaires à la compilation des modules et à la configuration du rôle *target* ?

```
# aptitude install iscsitarget iscsitarget-source module-assistant
```

3. Quelles sont les instructions de l'outil module-assistant qui permettent de préparer la compilation des modules et la mise à jour de la base des paquets ?

```
# m-a prepare -t
Récupération des sources du noyau de la version : 2.6.32-3-amd64
apt-get install linux-headers-2.6.32-3-amd64
```

```
# m-a update -t
```

[23] http://www.unixgarden.com/index.php/administration-reseau/le-support-du-protocole-iscsi-dans-linux
[24] http://wiki.debian.org/iSCSI
[25] http://sourceforge.net/mailarchive/message.php?msg_name=E2BB8074E5500C42984D980D4BD78EF9029E41AF%40MFG-NYC-EXCH2.mfg.prv
[1] http://wiki.debian.org/fr/ModuleAssistant

4. Quelle est la syntaxe de construction du paquet des modules `iscsitarget` ?

La commande de construction du paquet contenant le module est la suivante :

```
# m-a a-i iscsitarget -t
```

5. Comment installer le paquet construit avec l'outil module-assistant ?

Les paquets construits avec l'outil module-assistant sont toujours placé à la racine de l'arborescence des sources du noyau Linux ; soit le répertoire `/usr/src`.

```
# ls -goAh /usr/src/
total 100K
-rw-r--r-- 1  43K avril 28 20:49 iscsitarget-module-2.6.32-3-amd64_0.4.17+svn229-1.4+2.6.32-9_amd64.deb
-rw-r--r-- 1 3,7K avril 28 20:44 iscsi-target.patch
-rw-r--r-- 1  38K févr. 16 11:37 iscsitarget.tar.bz2
lrwxrwxrwx 1   37 avril 28 20:36 linux -> /usr/src/linux-headers-2.6.32-3-amd64
drwxr-xr-x 4 4,0K avril 28 20:29 linux-headers-2.6.32-3-amd64
drwxr-xr-x 4 4,0K avril 28 20:29 linux-headers-2.6.32-3-common
lrwxrwxrwx 1   26 avril 28 20:29 linux-kbuild-2.6.32 -> ../lib/linux-kbuild-2.6.32
lrwxrwxrwx 1   28 avril 28 20:30 linux-OLDVERSION.1272479765 -> linux-headers-2.6.32-3-amd64
drwxr-xr-x 3 4,0K févr. 16 11:37 modules
```

Comme il s'agit d'un paquet local au système, il s'installe à l'aide de la commande **dpkg**.

```
# dpkg -i iscsitarget-module-2.6.32-3-amd64_0.4.17+svn229-1.4+2.6.32-9_amd64.deb
```

On peut donc faire le bilan des paquets iSCSI installés sur le système ayant le rôle *target*.

```
# aptitude search ~iiscsi
i   iscsitarget                       - iSCSI Enterprise Target userland tools
i   iscsitarget-module-2.6.32-3-amd64  - iSCSI Enterprise Target module for Linux (kernel 2.6.32-3-amd64)
i   iscsitarget-source                - iSCSI Enterprise Target kernel module source
```