Solutions de stockages réseaux

Adrien Waksberg
Jonathan Demmerle
Sofiane El Harsal
Mohamed Benoikken

LP ASRALL
Année universitaire 2011-2012

Université Nancy 2
IUT Nancy-Charlemagne

Table des matières

Chapitre 1

Contexte du travail

De nos jours l'informatique a une place importante, si ce n'est indispensable pour les entreprises. L'informatique peut être utilisée pour la promotion de l'entreprise, la divulgation d'informations, et bien d'autres tâches, mais elles reposent toutes sur la gestion des données. Le stockage de ces données est devenu une préoccupation majeur pour le bon fonctionnement des entreprise, en effet les données peuvent être sensibles. Ainsi le stockage doit désormer allier perfommance de stockage, capacité de stockage, et sécurité des données.

FIGURE 1.1 – Baie du Grid'5000 à Nancy

1.1 Présentation du projet

Il existe différentes méthodes de stockage de données via le réseau, et on peut se demander quel sont les applications les plus adaptées pour chaque technologie.
Le but de notre projet est d'étudier différentes technologies de stockage réseau sur la plate-forme du Grid'5000. Cela consiste à les déployer sur des serveurs, et à tester leurs performances.

Durant ce projet nous avons étudié les technologies suivantes :

- NFS
- DRBD
- NBD
- iSCSI
- AoE

Dans une optique de performance et de sécurité nous allons coupler certaines de ces technologies avec la technologie de RAID.

1.2 Présentation du Grid'5000

Le Grid'5000 est un instrument de recherche pour les grilles et les systèmes pair-à-pair.
Il s'agit d'une plate-forme à grande échelle qui peut être facilement configurée, instrumentée et contrôlée.

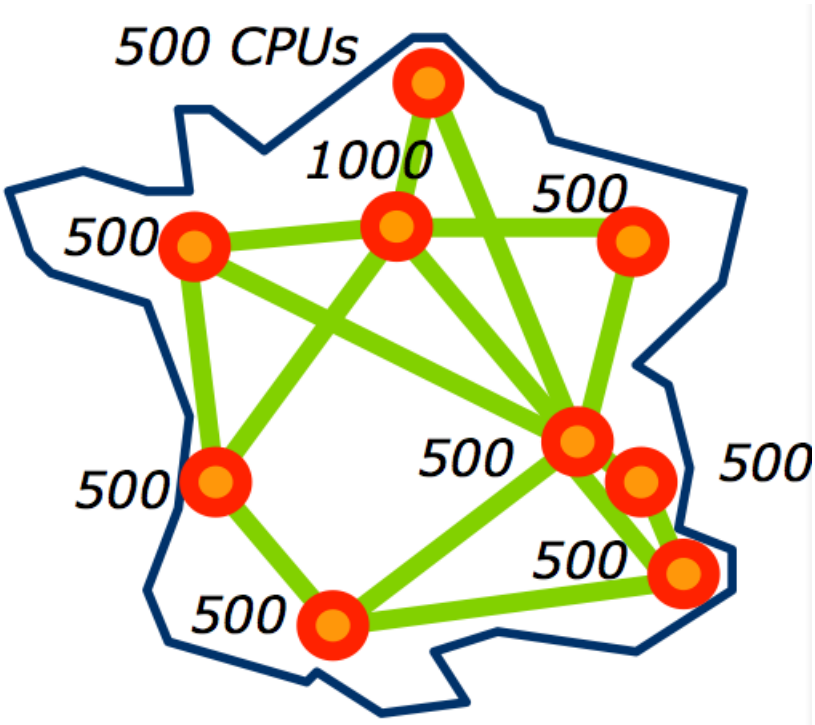


FIGURE 1.2 – Répartition géographique du Grid'5000

Grid'5000 vise a réunir 5000 processeurs sur plusieurs sites en France. Cette plate-forme est utilisée dans le cadre de projets expérimentaux, dans le domaine de la recherche dans les systèmes informatiques distribués et à grande échelle.

1.3 La plateforme

Dans le cadre de notre projet, nous avons pu visiter la plate-forme en présence de notre tuteur, et nous avons été autorisés à l'utiliser pour réaliser nos expériences.

Nous pouvons nous connecter à la plate-forme via le protocole ssh [1] avec la commande :

```
ssh login@access.grid5000.fr
```

On arrive sur un « frontend », un serveur d'accueil, général du Grid'5000. Depuis celui-ci on peut se connecter au site sur lequel on souhaite travailler, comme par exemple Nancy :

```
ssh nancy
```

Une fois cela fait, on se retrouve sur le frontend de Nancy.
Une session est créée et nous avons alors accès à un dossier personnel dans lequel nous allons typiquement placer les scripts dont nous avons besoin pour chaque déploiement [2].

```
scp monScript login@acces.grid5000.fr:nancy
```

L'utilisation de la plate-forme repose alors sur un fonctionnement simple :
Chaque personne souhaitant utiliser une machine, un nœud, doit en faire la demande en exécutant une réservation pour :

- un site choisi
- un nombre de nœud
- un moment
- et une durée donnée

Par exemple :

```
oarsub -r '2012-01-27 12:00:00' -l walltime=04:00:00 -t deploy nodes=2
```

revient à faire la demande de 2 nœuds, pour 4H, le 27 janvier à partir de midi.

1.4 API

La plate-forme grid'5000 met à disposition une API permettant de visualiser l'état des différents sites, de créer des jobs et d'en avoir la gestion, le tout via un script lancé localement sur votre machine après avoir tunnélisé une connexion via ssh.
Pour faire transiter les requêtes jusqu'à la plate-forme, l'api utilise le format Json.
Il est possible d'utiliser n'importe quel langage qui gère le format Json, mais nous avons opté pour ruby, qui est le langage vu cette année en cours.

1.4.1 Script

Le script créé permet de tunnéliser la connexion, de faire une réservation, de déployer une iso et de lancer un script externe qui s'occupera de faire la post installation.

1. Secure Shell
2. Un déploiement :
Cela consiste en fait à l'installation d'une distribution sur une machine mise à notre disposition.

Tunnel ssh

Pour que le script de réservation puisse créer un tunnel ssh il faut ajouter les lignes suivantes dans le fichier /.ssh/config :

```
1  Host g5k
2      User LOGIN_GRID5000
3      Hostname access.grid5000.fr
4      ForwardAgent no
5
6  Host *.grid5000.fr
7      User root
8      StrictHostKeyChecking no
9      UserKnownHostsFile=/dev/null
10     ProxyCommand ssh g5k "nc -q 0 `basename %h .g5k` %p"
11     ForwardAgent no
```

Installation

Attention avec des distributions anciennes n'ayant pas rubygems 1.8 ne pourront pas lancer le script. Tout d'abord installer ruby (1.8) et rubygems (1.8) :

```
# apt-get install ruby rubygems libreadline-dev
```

Installer le module restfully de ruby avec gem :

```
# gem install restfully
```

Utilisation

Lancement simple sans script de post installation :

```
ruby reserv.rb
```

Lancement simple avec un script de post installation :

```
ruby reserv.rb scritp.sh
```

Puis suivre les indications du script en précisant le site et le nombre d'heure que va durer la réservation. À la fin du déploiement de la l'iso le script en argument, sera lancé avec les adresses des nœuds alloués. Si le job n'est pas activé au bout de 5 minutes, cela peut être dû à un manque de noeuds libres, le scripts supprimera celui-ci.

Options

- -h, –help : affiche l'aide
- -s, –site : précise le site
- -t, –time : précise le temps de la réservation
- -n, –node : précise le nombre de noeuds à réserver
- -j, –job : précise le numéro d'un job existant
  Si celui ci est précisé le déployement se déroulera sur les noeuds alloués pour ce job
- -d, –delete : supprime un job existant, l'option -j doit être utilisé avec celle-ci
- -r, –reconnect : en cas de déconnexion, cette option permet de recréer le tunnel ssh
- -x, –xen : remplace l'iso d'installation par une comportant un kernel xen

Exemple d'une réservation de deux noeuds sur le site de nancy pendant 3 heures avec le déploiement d'un script :

```
ruby reserv.rb -n 2 -s nancy -t 3 scritp.sh
```

Logs

Des logs sont disponibles dans un fichier nommé « reserv-log.log ». Des informations comme l'heure de lancement, de déploiement et la fin d'exécution des scripts, ainsi que le numéro du job, le site et les noeuds alloués seront disponibles dans ce fichier de log.

Chapitre 2

Technologies de stockage

2.1 Technologie de partage de fichier

2.1.1 Présentation

Le partage de fichiers est une technique consistant à distribuer ou à donner accès, à distance, à des données numérisées (il peut d'agir de fichiers de toutes sortes : logiciels, livres, vidéos, audios etc...). Deux techniques de partage de fichiers existent actuellement :

- le pair-à-pair qui consiste à mettre des données en partage suivant un modèle de réseau informatique où chaque client est aussi un serveur
- l'hébergement centralisé qui permet de stocker les données sur un serveur de fichiers

2.1.2 NFS

NFS [1] est un système de fichier en réseau développé par Sun Microsystem, qui permet à des ordinateurs sous des systèmes UNIX de partager des données via le réseau.
La particularité de NFS est qu'il est simple à appréhender et à mettre en place.

La version 4 de NFS permet d'utiliser « Kerberos », qui est un système d'identification par clés secrètes et de tickets, pour l'identification sur le réseau, cette version gère aussi l'identification utilisé par la version 3.

Installation sur le serveur

Installer le paquet nfs-kernel-server :

```
# apt-get install nfs-kernel-server
```

Créer un dossier qui contiendra les données qui seront partagées avec les droits associés :

```
# mkdir /var/nfs
# chmod 777 /var/nfs
```

Ajouter dans le fichier */etc/export*, la ligne suivante, pour autoriser tout les ordinateurs présents sur le réseau, à lire et écrire dans le dossier partagé :

1. Network File System

```
/var/nfs        *(rw,sync,no_subtree_check)
```

Relancer le service :

```
# /etc/init.d/nfs-kernel-server reload
```

Installation sur les clients

Installer le paquet nfs-common :

```
# apt-get install nfs-common
```

Créer un dossier où sera monté le dossier partagé :

```
# mkdir /media/nfs
```

Monter le dossier distant :

```
# mount IP_SERVEUR:/var/nfs /media/nfs
```

Vous pouvez dès a présent ajouter, lire et modifier les fichiers présents sur le partage réseau.

2.2 Technologie type device

2.2.1 Présentation

Nous allons voir les différentes manières dont s'effectue l'export de « block devices [2] »à travers le réseau avec plusieurs technologies. Il existe différents outils, utilisant des technologies et des protocoles différents. Nous allons voir que la mise en place de ces outils s'effectue de manière plus ou moins complexe.

2.2.2 NBD

NBD [3] est un système qui permet de partager un « block device »via le réseau avec un autre système de type GNU/Linux.

Installation sur le serveur

Installer le paquet nbd-server :

```
# apt-get install nbd-server
```

Si vous avez une partition non allouée que vous voulez partager, il faut la formater :

```
# mkfs.ext4 /dev/sda2
```

Partager le périphérique dans notre cas :

```
# nbd-server 2000 /dev/sda2
```

2. périphérique bloc
3. Network Block Device

Installation sur le client

Installer le paquet nbd-client :

```
# apt-get install nbd-client
```

Créer un dossier où sera monté le dossier partagé :

```
# mkdir /media/nbd
```

Monter le dossier distant :

```
# nbd-client IP_SERVEUR 2000 /dev/nbd0
# mount /dev/nbd0 /media/nbd
```

2.2.3 iSCSI

SCSI [4] est un standard qui définit un bus informatique pour relier un ordinateur à un autre ordinateur (ou à un périphérique). Les sociétés Cisco et IBM ont développés ensemble un prototype de SCSI sur tcp/ip, ce qui a donné naissance à iSCSI en 1999.
Le protocole iSCSI est un protocole de la couche application permettant le transport de commandes SCSI. En 2001, IBM et Cisco ont chacun développé un moyen de stockage purement iSCSI.

Mise en place

Pour mettre en place une solution de stockage iSCSI, il faut au moins deux machines : un client et un serveur.

Installation sur le serveur :

- Installer les paquets iscsitarget et iscsitarget-dkms, permettant de créer les modules du noyau linux du paquet iscsitarget :

  ```
  apt-get install iscsitarget iscsitarget-dkms
  ```

- Démonter la partition que l'on souhaite partager :

  ```
  umount /tmp/
  ```

- Modifier le fichier de configuration */etc/iet/ietd.conf* :

  ```
  Target asrall.fr
  Lun 0 Path=/dev/sda5,Type=blockio
  ```

  Rédémarrer le service iscsitarget :

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

- Modifier le fichier de configuration */etc/default/iscstarget* :

  ```
  echo "ISCSITAGET_ENABLE = TRUE" > /etc/default/iscstarget
  ```

- Vérifier que le serveur est opérationnel. Tout d'abord, il faut vérifier que le démon ietd est bien lancé :

  ```
  ps -edf | grep ietd | grep -v grep
  root      4269     1  0 16:40 ?        00:00:00 /usr/sbin/ietd
  ```

4. Small Computer System Interface

Ensuite, vérifier que ce dernier est en écoute sur le port 3260 (configuration par défaut) :

```
netstat -anpt | grep LISTEN | grep 3260
tcp    0    0 0.0.0.0:3260    0.0.0.0:*    LISTEN    4269/ietd
```

Puis vérifier les volumes exportés par le serveur :

```
cat /proc/net/iet/volume
tid:1 name: asrall.fr
```

Récupèrer l'adresse IP, car on en aura besoin pour configurer le client :

```
ifconfig
```

Installation sur le client

- Installer le paquet open-iscsi :

  ```
  apt-get install open-iscsi
  ```

- Créer le repértoire où l'on montera la partition partagée (du serveur) :

  ```
  mkdir /media/partage
  ```

- Mettre le client en mode « decouverte », afin qu'il puisse connaître les « targets [5] »avec les details du serveur afin de pouvoir s'y connecter. Il a fallut au préalable récupérer l'adresser IP sur serveur (celle récupérée ici est : 172.16.14.8) :

  ```
  iscsiadm -m discovery --type sendtargets --portal=172.16.14.8
  ```

- Se connecter à la target :

  ```
  iscsiadm -m node -T asrall.fr -p -l 172.16.14.8
  ```

  Une fois la connexion établie, on vérifie les partitions disponibles et on récupère la nouvelle partition commençant par "/dev/sd". Dans notre cas la nouvelle partition est /dev/sdb

  ```
  cat /proc/partitions | grep -v sda | grep sd
  ```

- Formater la partition :

  ```
  mkfs.ext4 -q /dev/sdb
  ```

- Monter la partition dans /media/partage/ :

  ```
  mount /dev/sdb /media/partage
  ```

Voila de manière simple, comment configurer un système de stockage iSCSI entre un client et un serveur.
Le script bash correspondant est disponible dans la rubrique *Codes et fichiers*. Pour la configuration de iSCSI avec les systèmes de raid, nous avons utilisé un langage plus évolué (ruby) car nous avions besoin d'utiliser les tableaux associatifs permettant de faire correspondre les noms des serveurs à leurs adresses IP.

On a aussi la possibilité de mettre en place grâce à cette technologie un aspect un peu plus sécuritaire, qui consiste à implémenter un système d'authentification et aussi en restreignant l'accès par IP ou target. Pour l'authentification nous devons modifier le fichier de configuration du coté serveur, de la manière suivante :

5. cibles

```
Target asrall.fr
IncomingUser etudiant asrall
Lun 0 Path=/dev/mapper/iScsiShares_vg-iScsiShare01_lv,Type=fileio
```

Redémarrer le démon :

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

Coté client, modifier le fichier */etc/iscsi/iscsid.conf* :

```
node.session.auth.authmethod = CHAP
node.session.auth.username = etudiant
node.session.auth.password = asrall
```

Redémarrer le démon :

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

Voila comment on procède pour établir un système d'authentification.
Concernant la restriction IP/Target, il suffit de configurer le fichier */etc/initiators.deny*. Il existe un fichier de configuration */etc/initiators.allow* ; le fichier de restrictions est prioritaire sur le fichier d'autorisations. Donc on pourrait faire les choses de cette manière :

```
# Fichier : /etc/initiators.deny
ALL ALL
```

Ensuite :

```
# Fichier : /etc/initiators.allow
asrall.fr 172.16.14.9
```

Avec *asrall.fr* la « target »et *172.16.14.9* l'adresse IP du client autorisé à accéder à la « target ».

2.2.4 AoE

Présentation

Il faut avant tout savoir que AoE [6] est la solution la plus récente que nous étudions. Protocole de la couche réseau, il a été développé par the Brandtley Coile Compagny (ou Coraid Inc.) en 2005.

Principale concurente de la solution précédente : iSCSI, AoE offre de nombreux avantages par rapport à celui-ci par une plus grande simplicité et par une charge moins importante pour le processeur et le réseau grâce à l'utilisation de la couche MAC [7] pour les communications. De plus en plus reconnu par la communauté du libre, AoE a été récompensé lors du « LinuxWorld »de 2005 pour être la meilleur solution de stockage. De plus il faut noter le choix d'ouverture et son travail d'integration au noyau linux.

De part ses caractéristiques, AoE s'est trouvé être un choix judicieux dans la mise en place de SAN [8] à moindre frais.

6. ATA over Ethernet
7. Media Access Control
8. Storage Area Network

Détails techniques

Concrètement, AoE n'est rien de plus que le détournement de la norme SATA [9] transportant ses propres paquets via le réseau ethernet et la seconde couche du modèle OSI [10].

Présent au sein du noyau linux depuis sa version 2.6.11, on retrouve le support de AoE sur la plupart des systèmes d'exploitation modernes, tel que Mac OS X, Solaris, Windows, FreeBSD, OpenBSD, etc. On notera le fait que le driver [11] original soit sous licence GPL. Ce qui a permis à la communauté opensource d'en développer pour d'autres plate-formes, ce qui est le cas de Solaris et FreeBSD.

Suffisamment abouti, Coraid ne prévoit pas d'évolution immédiate du protocole AoE. En effet, du point de vue sécurité, AoE a de quoi faire rougir ses concurents. Tout d'abord, d'un point de vue réseau, AoE n'utilise pas le protocle internet (IP), on ne peut donc pas y accéder à partir d'internet ou d'une tout autre adresse IP présente sur le réseau. On peut plus rapprocher AoE d'un FCoE (Fibre Channel over Ethernet) que iSCSI ! À long terme, il s'agira de voir ce protocole s'implémenter dans les baies de stockage. AoE est non-routable : cela en fait un atout majeur niveau sécurité. Il n'existe pas de méchanisme de mot de passe ou de chiffrement.

D'un point de vue matériel, AoE s'en sort très bien. Si l'on imagine plusieurs accès en écriture sur le même périphérique, AoE peut assimiler diverses informations comme, entre autres, la source de la tentative d'écriture. Le protocole se chargera alors tout seul de décider à qui autoriser l'accès, tout en prévenant ceux mis en attente. Sans cela, beaucoup de fichiers se retrouveraient corrompus.

Mise en place

Lorsque l'on recherche des informations sur AoE, une des particularités souvent mentionnée est le fait que la documentation est divisée par environ 20 fois en comparaison à celle de iSCSI.

Lancer le service d'AoE sur le client et le serveur :

```
ssh root@$server modprobe aoe
```

Installation sur le serveur

Installer l'outil vblade spécifique au protocole AoE :

```
ssh root@$server apt-get install vblade -y --force-yes
```

Installation sur le client

Installer le paquet aoetools :

```
ssh root@$client apt-get install fio aoetools -y --force-yes
```

Créer le répertoire */media/partage* dans lequel on montera le système de fichier :

```
ssh root@$client mkdir /media/partage
```

Exporter sur le réseau une partition, ici /dev/sda5 :

```
ssh root@$server vblade-persist setup 0 1 eth0 /dev/sda5
ssh root@$server vblade-persist start 0 1
```

9. Serial Advanced Technology Attachment
10. Open Systems Interconnection
11. pilote

Il faut savoir que l'on peut très bien exporter d'autres types de systèmes de fichiers tels qu'une image ISO [12], un disque dur, une partition, ou encore un volume logique créé grâce à LVM.

Comme on peut le voir, des options sont précisées. En fait, chaque péripherique est défini par un couple de numéro unique. Ici par exemple, 0 (on imagine que cela correspond au numéro du serveur) et 1 (correspond quant à lui correspond au numéro de la partition).

On indique enfin sur quelle interface naviguer : ici, eth0. Cela peut apparaître annodin, mais on peut alors mettre en place plusieurs cartes réseau sur lesquelles transiteront les données, AoE comprendra alors de lui même, par exemple : losrqu'il y a une déteérioration d'un des câbles.
Il reste maintenant à accepter ce transfert sur le client.
Ces commandes permettront d'afficher divers informations :

```
ssh root@$client aoe-discover
ssh root@$client aoe-stat
```

Formater la partition monté automatiquement par aoetools :

```
ssh root@$client mkfs.ext4 /dev/etherd/e0.1
```

Monter la cible ainsi créée au répertoire /media/partage :

```
ssh root@$client mount /dev/etherd/e0.1 /media/partage
```

On peut alors directement l'utiliser.

2.2.5 DRBD

Principe

DRBD (pour "Distribued Replication Block Device", ou périphérique en mode bloc répliqué et distribué en français) est une solution logicielle pour une architecture de stockage distribuée permettant de synchroniser, par réplication, des périphériques de stockage entre des serveurs par le réseau. C'est une solution logicielle, elle utilisera les matériels standards déjà disponibles sur des serveurs. Pour implémenter DRBD, il faudra au minimum deux serveurs, avec, sur chacun d'entre eux, une connexion réseau.

Fonctionnement

Par analogie, on peut dire que DRBD s'apparente à du RAID1 via le réseau. Quand une écriture a lieu sur le disque du premier serveur, l'écriture est simultanément réalisée sur le second serveur. La synchronisation est faite au niveau de la partition et DRBD assure une cohérence des données au niveau du périphérique.
Cependant, il faut noter que DRBD n'est pas très sécurisé : il n'y a aucune authentification de contrôle d'accès, peu de confidentialité, et pas de cryptage sur des données échangées. On priviligera donc une utilisation sur des serveurs d'un même réseau local.

DRBD fonctionne en temps réel, c'est à dire qu'il tourne en tâche de fond alors que le serveur est disponible pour les applications. Les applications peuvent donc bien évidemment accèder aux données qui sont manipulées par DRBD. De plus, DRBD est complètement transparent pour les applications. Ceci est très utile, car toutes les applications peuvent fonctionner normalement sans configuration particulière. Les différents périphériques de type bloc utilisables pour DRBD sont très variés. En effet, DRBD permet l'utilisation de disques entiers, de partitions classiques, de partitions RAID, ou encore de volumes logiques

12. Organisation internationale de normalisation

LVM. Plusieurs modes sont disponibles pour le fonctionnement de DRBD au niveau de la réplication des données, ce qui permettra de jouer entre avec les temps de latence pour les applications et protection des données en cas de basculement. On pourra donc adapter DRBD à l'environnement réseau des serveurs et à l'utilisation qui leur en est faite.

En général, dans des systèmes de cluster conventionnels, on utilise un espace de stockage partagé pour que les données soient accessibles par les deux ressources du cluster. L'avantage de DRBD par rapport à ce type d'architecture est la redondance des espaces de stockage. En effet, avec ce type d'architecture, l'espace de stockage est un point individuel de défaillance. Autrement dit, la chute de l'espace de stockage partagé entraînera une indisponibilité pour les applications utilisant ces données. Avec DRBD ce problème n'apparait pas puisque même les espaces de stockage sont redondants. De plus, un split brain (déconnexion de deux parties d'un cluster) sur une architecture de ce type sera plus problématique que sur une architecture DRBD.

Mise en place

On a besoin d'un serveur primaire et d'un serveur secondaire pour mettre en place cette technologie. Sur les deux machines il faut installer le paquet drbd8-utils :

```
apt-get install drbd8-utils
```

Une fois installé, on s'occupe de la configuration. On place dans le répertoire */etc/drbd.d/* du primaire et du secondaire le fichier de configuration *.res suivant :

```
resource r0 {
        protocol C;

        startup {
                wfc-timeout 0;
                degr-wfc-timeout 120; # 2 minutes.
                        become-primary-on PRIMARY;
        }
        on PRIMARY {
                disk            /dev/sda5;
                address         IPP:7789;
                device          /dev/drbd0;
                meta-disk       internal;
        }
        on SECONDARY {
                disk            /dev/sda5;
                address         IPS:7789;
                device          /dev/drbd0;
                meta-disk       internal;
        }
}
```

Le *protocole C* est le protocole le plus sûr pour la protection des données. C'est aussi le plus couramment utilisé. Dans ce cas, le protocole de réplication est synchrone. En effet, l'opération d'écriture est considérée comme terminée lorsque les données on été écrites sur le disque local et que l'écriture sur le disque distant (du nœud pair) a été confirmée. Un basculement forcé n'engendrera normalement aucune perte de donnée. Ce protocole est certe le plus fiable, mais il n'empêche pas la perte de données. La réplication via DRBD, même avec le protocole C ne permet pas de se prémunir à 100%. En effet, rappelons que la perte

des données irréversiblement sur chacun deux membres au même moment induira inévitablement la perte totale des données. Nous pouvons voir dans ce fichier de configuration que le primaire et le secondaire sont configurés de la même manière.

Pour la suite, nous devons récupérer les adresses IP du primaire et du secondaire. Une fois cela effectué on effectue les configuration suivant :

```
sed -r "s/PRIMARY/$primary/g" -i /etc/drbd.d/r0.res
sed -r "s/SECONDARY/$secondary/g" -i /etc/drbd.d/r0.res
sed -r "s/IPP/$ip_primary/g" -i /etc/drbd.d/r0.res
sed -r "s/IPS/$ip_secondary/g" -i /etc/drbd.d/r0.res
```

Cela permet de changer le nom du serveur primaire et secondaire (exemple : granduc-16 par défaut) en *PRIMARY* et *SECONDARY*, ainsi que *IPP* et *IPS* par leurs adresses IP respectives. Démonter le */tmp/* :

```
umount /tmp
```

Formater la partition sda5 en *ext4* :

```
mkfs.ext4 /dev/sda5
```

Éffacer complétement les données présentes dans la partition :

```
e2fsck -f /dev/sda5 && resize2fs /dev/sda5 100G
resize2fs /dev/sda5 100G
```

Créer un bloc device avec l'outil drbdadm :

```
drbdadm create-md r0
```

Créer le répertoire de montage, où on monte la partition drbd sur ce répertoire (sur le serveur primaire) :

```
mkdir /media/partage
mount /dev/drbd0 /media/partage
```

Charger le module drbd (sur le primaire et secondaire) :

```
modprobe drbd
```

Monter le bloc device :

```
drbdadm up r0
```

Synchroniser le serveur primaire avec le serveur secondaire :

```
drbdadm -- --overwrite-data-of-peer primary r0
```

2.3 Le RAID

Le RAID [13] est une manière de répartir les données sur plusieurs disques durs afin de gagner en performance ou avoir une tolérance de panne plus grande.
À ses débuts le RAID permettait d'avoir de gros espaces de stockage avec des petits disques, ce qui était moins onéreux que d'acheter un gros disque dur. De nos jours le prix du Go [14] d'un disque dur étant devenu faible, on l'utilise surtout pour des questions de sécurité ou de performance.

13. Redundant Array of Independent Disks
14. Giga octet

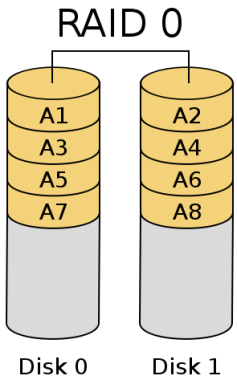
Le RAID a été défini par l'université de Berkeley en 1987, il peut être géré par la partie matériel ou logiciel d'un équipement informatique. Il existe deux types de RAID : le RAID matériel et le RAID logiciel. Le RAID matériel est géré par un composant physique et est plus performant. Le RAID logiciel est géré par le système d'exploitation qui regroupe plusieurs partitions avec des disques différents ; Le CPU fait tous les calculs en utilisant la RAM centrale.

2.3.1 RAID0

Le RAID0 permet d'≪ entrelacer ≫plusieurs disques durs, afin d'avoir un disque dur du total de la sommes des disques.

Avantages :
- grand espace de stockage
- vitesse d'écriture accrue

Inconvénients :
- perte totale des données en cas de crash d'un disque

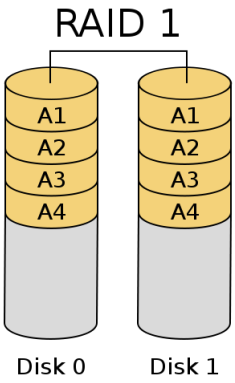


2.3.2 RAID1

Le RAID1 permet de faire de la redondance entre plusieurs disques ce qui permet une plus grande fiabilité.

Avantages :
- les données sont sauvées tant qu'un disque est toujours opérationnel
- vitesse de lecture améliorée

Inconvénients :
- le coût est proportionnel au nombre de mirroir
- l'espace de stockage est égale au plus petit volume

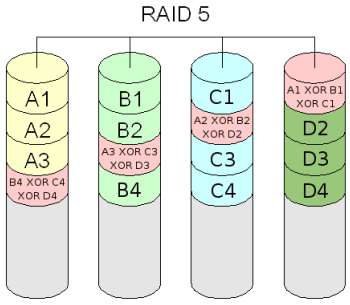


2.3.3 RAID5

Le RAID5 est un compromis entre sécurité des données et espace disque disponible. Pour cela des données de parités sont réparties sur chaque disque et permettent, en cas de crash d'un disque, de reconstituer celui-ci grâce à ses données ainsi que celles présentes sur les autres disques.

Avantages :
- surcoût minimal, un disque de plus que du RAID0
- vitesse de lecture équivalente au RAID0
- aucune perte de données lors du crash d'un disque

Inconvénients :
- vitesse d'écriture réduite
- minimum de 3 disques
- la syncrhonisation d'un nouveau disque après le remplacement d'un ancien peut prendre beaucoup de temps



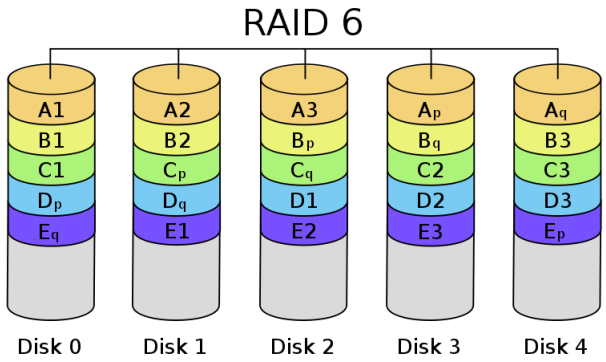
2.3.4 RAID6

Le RAID6 est une évolution du RAID5 qui permet d'améliorer la sécurité des données en ayant plusieurs copies des données redondantes au lieu d'une seule.

Avantages :
- vitesse de lecture équivalente au RAID0

- aucune perte de données lors du crash de deux disques

Inconvénients :
- vitesse de lecture réduite
- minimum de 4 disques
- la syncrhonisation d'un nouveau disque après le remplacement d'un ancien peut prendre beaucoup de temps

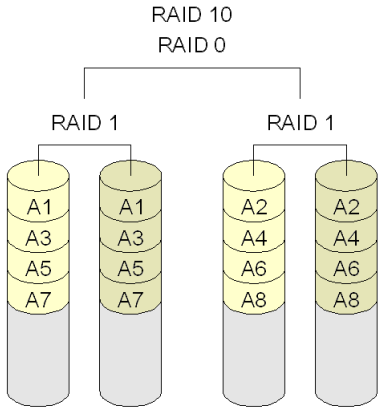


2.3.5 RAID10

Le RAID10 est la combinaison de deux RAID1 (ou plus) avec du RAID0.

Avantages :
- temps de reconstruction rapide
- vitesse de lecture et écriture accrue
- aucune perte de données tant qu'un disque est opérationnel sur chaque grappe

Inconvénients :
- minimum de 4 disques
- coût

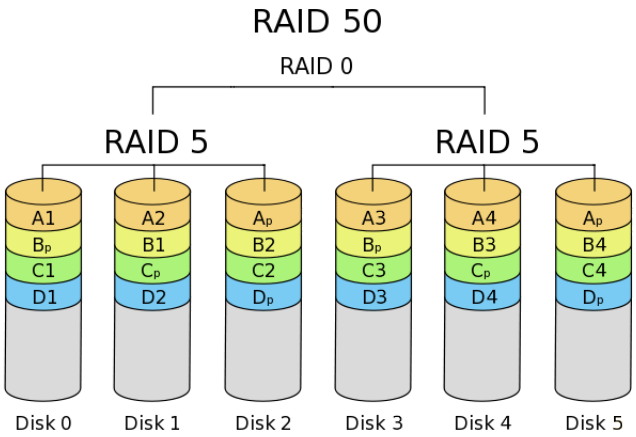


2.3.6 RAID50

Le RAID50 est la combinaison de deux RAID5 (ou plus) avec du RAID0.

Avantages :
- temps de reconstruction rapide
- vitesse de lecture et écriture accrue
- aucune perte de donnée si un disque lâche sur une grappe

Inconvénients :
- minimum de 6 disques
- coût



2.3.7 Outil RAID logiciel : mdadm

mdadm est un utilitaire standard sous linux permettant de gérer, créer, assembler des RAID logiciels. Il faut tout d'abord créer un fichier spécial de type bloc :

```
mknod /dev/md0 b 9 0
```

Pour créer un raid 0, 1, 5, 6 ou 10 on procède de la même manière. Par exemple, pour le RAID0, il suffit de faire :

```
mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb /dev/sdc
```

L'option *–level* indique le niveau du raid (pour du raid 5 on aurait mis *–level=5*). L'option *–raid-devices* indique quant à elle le nombre de bloc pour la création du RAID.
Il faut savoir qu'on a besoin d'au moins :
- 2 disques pour du RAID0 et du RAID1
- 3 disques pour le RAID 5
- 5 disuqes pour le RAID 6
- 4 disques pour le RAID 10
- 6 disques pour le raid 50

/dev/sdb et */dev/sdc*) correspondent aux partitions existantes sur le client.

Remarque : On incrèmente de 1 le nombre de nœuds à réserver à chaque fois (il ne faut pas oublier le client).
Une fois cela effectué, on formate la partition et on la monte dans un repértoire :

```
mkdir /media/partage
# On formate la partition (en ext4 dans notre cas)
mkfs.ext4 /dev/md0
# On monte la partition
mount /dev/md0 /media/partage
```

Comme nous l'avons dit plus haut, pour le RAID 50, il faut assembler deux RAID 5 (minimum) avec un RAID 0. Pour cela on configure les choses de cette manière :

```
# Création des fichiers spéciaux de type bloc
mknod /dev/md0 b 9 0
mknod /dev/md1 b 9 0
mknod /dev/md50 b 9 0
# On créé les deux RAID 5
mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sdb /dev/sdc /dev
    /sdd
mdadm --create /dev/md1 --level=5 --raid-devices=3 /dev/sde /dev/sdf /dev
    /sdg
# On créé le RAID 50 avec les deux RAID 5
mdadm --create /dev/md50 --level=0 --raid-devices=3 /dev/md0 /dev/md1
# On formate la partition
mkfs.ext4 /dev/md50
# Puis on monte la partition
mount /dev/md50 /media/partage
```

Voila comment on effectue du RAID 50. Une fois qu'on a créé un RAID il peut nous arriver de faire une mauvaise manipulation ou de redémarrer la machine cliente, dans ce cas on perd le montage des disques. Pour résoudre ce petit problème on a créé des scripts de réassemblage de RAID pour chaque technologie (voir les *scripts de réassemblage*). Pour réaliser cette opération, il suffit d'utiliser la commande *mdadm –assemble*

```
mdadm --assemble /dev/md0 /dev/sdb /dev/sdc
```

Bien sûr, pour un RAID 50 il faudra réassembler les RAID 5, et appliquer un RAID 0 à ces deux RAID 5 :

```
mknod /dev/md0 b 9 0
mknod /dev/md1 b 9 0
mknod /dev/md50 b 9 0
mdadm --assemble /dev/md0 /dev/sdb /dev/sdc /dev/sdd
mdadm --assemble /dev/md1 /dev/sde /dev/sdf /dev/sdg
mdadm --assemble /dev/md50 /dev/md0 /dev/md1
mount /dev/md50 /media/partage
```

Chapitre 3

Évaluation

3.1 Fio

Pour réaliser nos tests nous avons dû utiliser un outil dit de benchmark. En français, band d'essai, qui permet de mesurer les performances (ou d'effectuer un stress test) d'un système. Au vue des tests que nous avons besoin d'effectuer, Fio était tout désigné.
Il permet, selon les commandes appropriées, d'effectuer des tests d'écritures et de lecture, de fichiers plus ou moins volumineux.
Une fois un environnement approprié déployé, typiquement un client et un serveur, il nous faut installer fio :

```
apt-get install fio
```

Nous avons dans notre cas installé à chaque fois une debian squeeze et donc nous travaillons avec la version 1.38 de Fio. Quatres fichiers test bien distincts ont alors été réalisés, regroupant chacun différents paramètres spécifiques tels que les notions d'aléatoire ou de séquentiel, ou encore le fait que le test soient réalisés avec un ou plusieurs fichiers.
Un fichier fio se compose alors comme ceci :

```
1 [global]
2 ioengine=sync
3 rw=randwrite                # Opération à effectuer
4 nrfiles=1                   # Nombre de fichier
5 size=40000m                 # Taille des fichiers
6 directory=/media/partage/   # Emplacement à tester
7
8 [job1]
```

3.2 Procédure

Les tests ont été réalisés sur le site du Luxembourg du Grid'5000 pour la plupart afin que les résultats obtenus soient le plus juste possible. Étant donné que le luxembourg était limité à 22 nœuds et que certains tests pouvaient durer jusqu'à 15 heures (voire plus), nous avons été contraints d'utiliser des nœuds sur Nancy (pour les RAID 50 notamment). Pour chaque technologie et chaque cas, nous les avons déployés avec nos scripts puis lancé 4 tests avec fio :

- une écriture aléatoire
- une écriture séquentielle
- une lecture séquentielle
- une lecture aléatoire

Chaque test a été réalisé avec un unique fichier de 40Go pour être sûr que le serveur ne puisse pas tout stocker temporairement dans la mémoire vive. De plus, toutes les partitions utilisées durant les tests, furent formatées en ext4.

3.3 Résultats

		Écriture		Lecture	
		Séquentielle	Aléatoire	Séquentielle	Aléatoire
Disque dur interne		73863	1952	85199	715
NFS		22993	26185	18766	491
DRBD		39597	NC	413829	7734
NBD	Simple	81930	4378	81280	883
	RAID 0	39028	1588	91910	1588
	RAID 1	25339	3095	72718	951
	RAID 5	31692	4657	63692	1297
	RAID 6	36610	3589	106544	9546
	RAID 10	60921	4581	102894	1564
	RAID 50	NC	NC	NC	NC
iSCSI	Simple	84978	1217	78961	686
	RAID 0	118093	1709	113537	762
	RAID 1	59283	904	78267	638
	RAID 5	53065	843	82181	688
	RAID 6	43768	1515	101030	752
	RAID 10	33373	1326	99730	
	RAID 50	43474	2121	68652	416
AoE	Simple	6132	3361	12279	662
	RAID 0	120716	7703	29077	768
	RAID 1	38998	3487	51782	801
	RAID 5	21195	1809	58772	714
	RAID 6	55976	4832	103299	13163
	RAID 10	16929	2201	47315	893
	RAID 50	NC	NC	NC	13684

FIGURE 3.1 – Toutes les valeurs sont en ko/s

3.4 Interprétation des résultats

Tout d'abord il faut savoir que les résultats en rouge sont forcément erronés et que ceux en orange le sont sans doute aussi. En dehors des résultats qui nous semble sortir de nul part, sans doute dûs à un problème de disque dur sur certains serveurs du site du Luxembourg, et n'ayant pas le temps de refaire les tests nous pouvons constater que :

- le système de fichier partagé NFS est bien plus lent que les technologies de « bloc device »
- les lectures et écritures séquentielles sont plus rapides que les lectures et écritures aléatoires. Ceci s'explique peut-être par le fait que la tête de lecture du disque qui doit parcourir plus de chemin en aléatoire.
- les lectures aléatoires sont bien plus lentes que les écritures aléatoires, nous supposons que le temps de recherche de la prochaine adresse mémoire à lire est plus important que lors de l'écriture de cette dernière.

- pour les technologies de « bloc device »les performances sont à peu près équivalentes, ce qui peut réelement jouer est donc la vitesse du réseau.
- certaines valeurs obtenues pour certains type de RAID sont moins grandes que celles espérées, limitées par la vitesse du réseau (120Mo). Par exemple le RAID0 devrait aller au double de la vitesse du disque dur de référence soit environs 150Mo/s
- mais on constate aussi qu'en RAID certaines valeurs sont plus grandes que celles espérées car, en passant par différents serveurs, on a des buffers plus grand pour stocker les données temporairement dans la mémoire vive. On peut le constater pour le RAID 6 de NBD qui devrait normalement aller moins vite en lecture aléatoire que le disque de référence.

3.5 Utilisation

Pour le cas d'un simple mise à disposition des documents au sein d'une PME [1] il est conseillé d'utiliser une technologie de partage de fichier comme NFS, qui a l'avantage d'être très simple à mettre en place.

Les technologies de « bloc device »sont intéressantes dans le cas de création de SAN.

NBD est la solution la plus simple à mettre en place mais l'inconvénient est de ne pas avoir de sécurité lors de l'échange des données.
Son utilisation est recommandée pour un réseau local protégé.

AoE ajoute une sécurité par rapport à NBD mais ne peut pas être utilisé hors d'un réseau local.
Son utilisation est recommandée dans un réseau local peu protégé.

iSCSI est bien plus complexe que les deux précédents, mais permet de faire transiter les données via internet en toute sécurité.
Son utilisation est recommandée lorsque le disque distant n'est pas dans le réseau local.

Coupler certaines de ces technologies avec du RAID peut être bénéfique en terme :

- d'espace de stockage avec le RAID0
- de performances :
  - amélioration des écritures avec du RAID0
  - amélioration des lectures avec du RAID1
- de sécurité avec la redondance des données avec du RAID1
  On peut faire du RAID1 entre deux baies de stockages ce qui permet en cas de problème sur l'une d'entre elles, de toujours avoir accès aux données. Avec iSCSI on peut faire en sorte que ces deux baies soient situées dans des endroits complétement différents.
- de compromis entre l'espace de stockage, les performances et la sécurité grâce à tous les autres types de RAID qui peuvent exister.

1. Petite et Moyenne Entreprise

Chapitre 4

Organisation du projet

4.1 Répartition des tâches

Mohamed a semble-t-il abandonné la licence, il n'a pas vraiment participé au projet.
Jonathan a fait les prémices de AoE, DRBD, fio ainsi que les rapports.
Sofiane a réalisé les scripts pour iSCSI, des scripts de réassemblage des RAID, fio, a participé l'élaboration des rapports, et à la réalisation des tests.
Adrien a réalisé les scripts pour AoE, DRBD, NBD, l'API, des scripts de réassemblage des RAID, a participé à l'élaboration des rapports, et à la réalisation des tests.
L'étude de NFS et son script, ainsi que l'exploitation générale des résultats, ont quant à eux été réalisés en commun.

4.2 Difficultés rencontrées

Suite à un manque d'organisation au début du projet, nous avons perdu énormément de temps avec la prise en main de Fio étant donné que nous avions des valeurs biaisées puisque les données étaient temporairement stockées dans la mémoire vive du serveur.

Le problème majeur durant la phase de test était surtout dû au fait que certains tests, dont les lectures aléatoires, pouvaient prendre plus de 10h pour donner un résultat convenable. De plus, nous avions un manque de matériel disponible sur le site du luxembourg avec 22 machines qui pouvaient être allouées à d'autres personnes.
De ce fait, la difficulté principale du projet s'est ressentie plus au niveau du manque de matériel et de la gestion du temps, qu'à celui de la compréhension ou, encore, de la msie en place.

Certains tests n'ont pu être réalisés à cause d'un administrateur du grid'5000 qui a fait redémarrer les machines alors que des tests étaient en cours.

Chapitre 5

Conclusion

5.1 Conclusion générale du projet

Le projet en lui même était intéressant que ce soit au niveau de la plate-forme, ou encore du but recherché par le sujet. Nous avons pu travailler sur une grande infrastructure ce qui nous a permis de mettre en place plusieurs technologies sur un nombre de machine souhaité, sans avoir besoin de les avoir présentes physiquement. Cependant, le défaut qu'on peut lui repprocher est le manque d'un contexte réel qui pourrait arriver durant notre vie profesionnelle, l'idéal aurait été, par exemple, une mise en situation. Nous avons presque pu arriver à terme de nos objectifs souhaités malgré les difficultés rencontrées et les tests que nous n'avons pu approfondir.
Il faut maintenant bien se rendre compte qu'en terme de recherche et/ou d'utilité, une telle étude devrait mobiliser une equipe, et plus de materiel spécifique, sur une durée beaucoup plus longue.
Au final, ce que nous pouvons resortir de notre travail est qu'il existe bien des technologies ayant toutes leurs avantages et leurs inconvenients. À l'heure actuelle il n'a pas encore été découvert de panacée pour n'importe quel domaine de l'informatique et, à cause (ou grâce) de la rapidité d'évolution d'un point vue materiel ainsi que logiciel, il n'en existera peut-être jamais.

5.2 Expérience acquise

D'une manière générale nous nous sommes améliorés dans l'écriture de scripts en bash et ruby et à l'automatisation des tâches et des procédures de configuration que l'on a à effectuer.
De plus, nous avons maintenant de bonnes connaissances dans les technologies de RAID, mais aussi dans les technologies de stockages réseaux que nous avons étudiées. Ce projet nous a aussi permis d'appréhender un problème et de mettre en place une ou plusieurs solutions techniques liées à celui-ci.

5.3 Remerciements

Nous tenons à remercier notre tuteur Lucas Nussbaum qui a su nous guider malgré un début difficile et qui nous a aidé lorsque nous étions bloqués. Nous remercions aussi toute l'équipe du Grid5000 qui nous a permis d'avoir une plâteforme de test performante, sans quoi nous aurrions eu du mal à réaliser ce projet.

Chapitre 6

Bibliographie

Les manuels, lorsqu'ils étaient conséquents, ainsi que la page wikipedia (anglais), de chaque technologies ont bien entendu été consultés.

- https://www.grid5000.fr/mediawiki/index.php/Category:Portal:Tutorial
- https://api.grid5000.fr/
- http://www.horoa.net/2011/08/aoe-une-alternative-san-plus-que-credible/
- http://www.drbd.org/docs/introduction/
- http://fr.wikipedia.org/wiki/Advanced_Technology_Attachment
- http://fr.wikipedia.org/wiki/SCSI
- http://www.unixgarden.com/index.php/gnu-linux-magazine/ubuntuserveur-iscsi+
- http://sys-admin.wikidot.com/install-iscsi-target
- http://fr.wikipedia.org/wiki/RAID_%28informatique%29
- http://www.unixgarden.com/index.php/gnu-linux-magazine/ubuntuserveur-iscsi
- http://sys-admin.wikidot.com/install-iscsi-target
- http://fr.wikipedia.org/wiki/Kerberos

Chapitre 7

Annexes

7.1 Codes et fichiers

7.1.1 API

```
1  # Configure le ~/.ssh/config
2  # Lancer avec 'ruby reserv.rb'
3
4  require 'rubygems'
5  require 'restfully'
6
7  puts "\n#############################################"
8  puts "# Grid5000 API"
9  puts "#############################################\n\n"
10
11 # Ouverture du fichier de log
12 file = "reserv-log.log"
13 log = File.open(file, "a+")
14
15 # Gestion des arguments
16 i = 0
17 scripts = []
18 pid_ssh = `ps aux | grep "ssh -NL 3443:api.grid5000.fr:443 g5k" | grep -v
       "grep" | awk '{ print $2}'`
19
20 while i < ARGV.length do
21
22     case ARGV[i]
23         when '-h', '--help'
24             puts "Aide:"
25             puts "-h, --help"
26             puts "Affiche cette aide"
27             puts "-j, --job"
28             puts "Numero du job existant pour redeployer un OS"
```

```
29              puts "-s, --site"
30              puts "Nom du site (ex: nancy)"
31              puts "-n, --nodes"
32              puts "Nombre de noeuds a allouer"
33              puts "-t, --time"
34              puts "Temps de la reservation"
35              puts "-d, --delete"
36              puts "Supprime un job indique avec l'option -j"
37              puts "-r, --reconnect"
38              puts "Pour recreer le tunnel ssh, en cas de deconnexion"
39              puts "-x, --xen"
40              puts "Pour utiliser une image contenant xen"
41              exit 0
42          when '-r', '--reconnect'
43              system("kill #{pid_ssh}")
44              system('ssh -NL 3443:api.grid5000.fr:443 g5k &')
45              puts "Ouverture d'un nouveau tunnel ssh!"
46              exit 0
47          when '-d', '--delete'
48              delete = true
49          when '-j', '--job'
50              i += 1
51              job_uid = ARGV[i].to_sym
52          when '-s', '--site'
53              i += 1
54              site = ARGV[i]
55          when '-n', '--nodes'
56              i += 1
57              nodes = ARGV[i].to_i
58          when '-t', '--time'
59              i += 1
60              hours = ARGV[i]
61          when  '-x', '--xen'
62              image = "squeeze-x64-xen"
63          else
64              scripts << ARGV[i]
65      end
66
67      i += 1
68
69  end
70
71  # Ouverture d'un tunnel ssh
72  if pid_ssh.chomp.empty?
73      system('ssh -NL 3443:api.grid5000.fr:443 g5k &')
74      puts "Ouverture d'un nouveau tunnel ssh!"
75      sleep 3
76  end
77
```

```
78  logger = Logger.new(STDERR)
79  logger.level = Logger::WARN
80  Restfully::Session.new(:logger => logger, :base_uri => 'https://localhost
        :3443/2.1/grid5000') do |root, session|
81  begin
82
83
84      # Formulaire de reservation
85      if site.nil?
86          i = 0
87          sites = []
88          puts "Liste des sites:"
89          root.sites.each { |value|
90              sites << value['uid']
91              puts "##{i} #{value['uid']}"
92              i += 1
93          }
94          puts "\n"
95
96          print "Lieu de la reservation (entrez le numero du site): "
97          number_site = STDIN.gets.chomp.to_i
98          site = sites[number_site]
99      end
100
101     if job_uid.nil?
102
103         if nodes.nil?
104             print "Nombre de noeuds a reserver: "
105             nodes = STDIN.gets.chomp.to_i
106         end
107         if hours.nil?
108             print "Duree de la reservation en heure: "
109             hours = STDIN.gets.chomp
110         end
111
112         # Configuration du job
113         job_to_submit = {
114                         :resources => "nodes=#{nodes},walltime=#{hours}",
115                         :command => "sleep #{3600 * hours.to_i}",
116                         :reservation => "#{Time.now.to_i}",
117                         :types => ["deploy"],
118                         :name => 'Stockages reseaux',
119                         :project => 'Stockages reseaux',
120         }
121
122         puts "*** Envois de la requete..."
123         job = root.sites[site.to_sym].jobs.submit(job_to_submit)
124         log.write("[#{Time.now.strftime("%Y-%m-%d %T")}] Creation du job
            ##{job['uid']} sur le  site #{site}\n")
```

```
125         puts "Attente du lancement du job ##{job['uid']} !"
126         timeout = Time.now.to_i+300
127         while job.reload['state'] != 'running'
128             print "."
129             STDOUT.flush
130             sleep 2
131             # Si le delais depasse 5min on supprime le job
132             if  Time.now.to_i >= timeout
133                 job.delete
134                 puts "\n"
135                 puts "/!\\ Le delais d'attente du debut du job est expire
                        !"
136                 puts "Le job a ete supprime!"
137                 log.write("[#{Time.now.strftime("%Y-%m-%d %T")}] Job ##{
                        job['uid']} supprime sur le site #{site}\n")
138                 exit 1
139             end
140         end
141     elsif !delete.nil?
142         root.sites[site.to_sym].jobs[job_uid].delete
143         puts "Job supprime!"
144         exit 0
145     else
146         job = root.sites[site.to_sym].jobs[job_uid]
147         job.reload
148     end
149     puts "\n"
150
151     # Configuration du type de deployement
152     if image.nil?
153         image = "squeeze-x64-base"
154     end
155     deployment_to_submit = {
156         :nodes => job['assigned_nodes'],
157         #:environment => 'http://public.nancy.grid5000.fr/~awaksberg/
                mywheezy.env',
158         :environment => "#{image}",
159         :key => "#{IO.read("#{ENV['HOME']}/.ssh/id_rsa.pub").chomp}"
160     }
161
162     puts "*** Debut du deploiment..."
163     log.write("[#{Time.now.strftime("%Y-%m-%d %T")}] Lancement du
            deploiement du job ##{job['uid']}\n")
164     log.write("Noeuds assignes au job ##{job['uid']}: #{job['
            assigned_nodes'].join(" ")}\n")
165     puts "Patientez durant le deploiement!"
166     deployment = root.sites[site.to_sym].deployments.submit(
            deployment_to_submit)
167     while deployment.reload['status'] == 'processing'
```

```
168     print "."
169         STDOUT.flush
170         sleep 5
171     end
172     puts "\n\n"
173 
174     rescue => e
175         puts "Erreur: #{e.class.name}, #{e.message}"
176         exit 1
177     end
178 
179     # Lancement d'un script avec les nodes en parametre
180     puts "*** Lancement des scripts de post-installation..."
181     log.write("[#{Time.now.strftime("%Y-%m-%d %T")}] Lancement des
           scripts postinstall du job ##{job['uid']} sur le site #{site}\n")
182     log.write("Scripts du job ##{job['uid']} du site #{site}: #{scripts.
           join(" ")}\n")
183 
184     scripts.each { |script|
185             ext = script.gsub(/^.*\.([^.]*)$/, '\1')
186             case ext
187                 when 'sh'
188                 system("sh #{script} #{job['assigned_nodes'].join(" ")}")
189                 when 'rb'
190                 system("ruby #{script} #{job['assigned_nodes'].join(" ")}
                       ")
191                 when 'py'
192                 system("python #{script} #{job['assigned_nodes'].join(" "
                       )}")
193                 else
194                 puts "Extension inconnue pour le script #{script},
                       impossible de lancer le script post installation!"
195                 exit 1
196             end
197     }
198 
199     puts "Numero du job ##{job['uid']}"
200     puts job['assigned_nodes'].join(" ")
201     log.write("[#{Time.now.strftime("%Y-%m-%d %T")}] Fin du deploiement
           du job ##{job['uid']} sur le site #{site}\n")
202 
203 end
204 
205 log.close
206 
207 exit 0
```

7.1.2 NBD

Simple

```
1  #!/bin/sh
2  # Script d'installation de NBD
3  # A utiliser qu'avec 2 noeuds
4  #-------------------------------------
5
6  echo "#------------------------------"
7  echo "# Deploiement de NBD"
8  echo "#------------------------------"
9
10 server=$1
11 client=$2
12
13 #---------------------------
14 # Configuration du serveur
15 #---------------------------
16
17 scp $HOME/.vimrc root@$server:/root/
18 ssh root@$server apt-get update
19 ssh root@$server apt-get install nbd-server -y --force-yes
20 ssh root@$server umount /tmp
21 ssh root@$server mkfs.ext4 /dev/sda5
22 ssh root@$server nbd-server 2000 /dev/sda5
23
24 #---------------------------
25 # Configuration des clients
26 #---------------------------
27
28 scp $HOME/.vimrc root@$client:/root/
29 scp $PWD/fio/*.fio root@$client:/root/
30 ssh root@$client apt-get update
31 ssh root@$client apt-get install fio nbd-client -y --force-yes
32 ssh root@$client mkdir /media/partage
33 ssh root@$client nbd-client $server 2000 /dev/nbd0
34 ssh root@$client mount /dev/nbd0 /media/partage
35
36 #---------------------------
37 # Affichage des informations
38 #---------------------------
39
40 echo "\033[01;33mNoeuds alloues\033[00m"
41 echo "Serveur: \033[01;31m$server\033[00m"
42 echo "Client: \033[01;34m$client\033[00m"
43
44 exit 0
```

RAID0

```
#!/bin/sh
# Script d'installation de NBD
# À utilisé qu'avec 3 noeuds ou plus
#-------------------------------------

echo "#------------------------------"
echo "# Deploiement de NBD (RAID0)"
echo "#------------------------------"

client=$1
shift
servers=$*

#---------------------------
# Configuration des serveurs
#---------------------------

for server in $servers; do
    scp $HOME/.vimrc root@$server:/root/
    ssh root@$server apt-get update
    ssh root@$server apt-get install nbd-server -y --force-yes
    ssh root@$server umount /tmp
    ssh root@$server nbd-server 2000 /dev/sda5
done

#---------------------------
# Configuration du client
#---------------------------

i=0
devices=""

scp $HOME/.vimrc root@$client:/root/
scp $PWD/fio/*.fio root@$client:/root/
ssh root@$client apt-get update
echo "all\n" | ssh root@$client apt-get install fio mdadm nbd-client -y
   --force-yes
ssh root@$client mkdir /media/partage

for server in $servers; do
    ssh root@$client nbd-client $server 2000 /dev/nbd$i
    devices="/dev/nbd$i $devices"
    i=$(($i+1))
done

ssh root@$client mknod /dev/md0 b 9 0
```

```
46 echo "y\n" | ssh root@$client mdadm --create /dev/md0  --level=0 --raid-
       devices=$i $devices
47 ssh root@$client mkfs.ext4 /dev/md0
48 ssh root@$client mount /dev/md0 /media/partage
49
50 #---------------------------
51 # Affichage des informations
52 #---------------------------
53
54 echo "\033[01;33mNoeuds alloues\033[00m"
55 echo "Serveurs: \033[01;31m$servers\033[00m"
56 echo "Client: \033[01;34m$client\033[00m"
57
58 exit 0
```

RAID50

```
#!/bin/sh
# Script d'installation de NBD
# À utilisé qu'avec 7 noeuds ou plus
#-------------------------------------

echo "#------------------------------"
echo "# Deploiement de NBD (RAID50)"
echo "#------------------------------"

client=$1
shift
servers=$*

#---------------------------
# Configuration des serveurs
#---------------------------

for server in $servers; do
    scp $HOME/.vimrc root@$server:/root/
    ssh root@$server apt-get update
    ssh root@$server apt-get install nbd-server -y --force-yes
    ssh root@$server umount /tmp
    ssh root@$server nbd-server 2000 /dev/sda5
done

#---------------------------
# Configuration du client
#---------------------------

i=0
devices_0=""
devices_1=""

scp $HOME/.vimrc root@$client:/root/
ssh root@$client apt-get update
echo "all\n" | ssh root@$client apt-get install fio mdadm nbd-client -y
   --force-yes
ssh root@$client mkdir /media/partage

for server in $servers; do
    ssh root@$client nbd-client $server 2000 /dev/nbd$i
    if [ $i -lt 3 ]; then
        devices_0="/dev/nbd$i $devices_0"
    else
        devices_1="/dev/nbd$i $devices_1"
    fi
    i=$(($i+1))
```

```
47 done
48
49 ssh root@$client mknod /dev/md0 b 9 0
50 ssh root@$client mknod /dev/md1 b 9 0
51 ssh root@$client mknod /dev/md50 b 9 0
52 echo "y\n" | ssh root@$client mdadm --create /dev/md0 --level=5 --raid-
      devices=3 $devices_0
53 echo "y\n" | ssh root@$client mdadm --create /dev/md1 --level=5 --raid-
      devices=3 $devices_1
54 echo "y\n" | ssh root@$client mdadm --create /dev/md50 --level=0 --raid-
      devices=2 /dev/md0 /dev/md1
55 ssh root@$client mkfs.ext4 /dev/md50
56 ssh root@$client mount /dev/md50 /media/partage
57
58 #----------------------------
59 # Affichage des informations
60 #----------------------------
61
62 echo "\033[01;33mNoeuds alloues\033[00m"
63 echo "Serveurs: \033[01;31m$servers\033[00m"
64 echo "Client: \033[01;34m$client\033[00m"
65
66 exit 0
```

7.1.3 iSCSI

Simple

```
#!/bin/sh
# Script d'installation d'iSCSI
# À utiliser qu'avec 2 noeuds
#--------------------------------------

echo "#------------------------------"
echo "# Deploiement de iSCSI"
echo "#------------------------------"

server=$1
client=$2

#---------------------------
# Configuration du serveur
#---------------------------

scp $HOME/.vimrc root@$server:/root/
ssh root@$server umount /tmp
ssh root@$server apt-get update
ssh root@$server apt-get install iscsitarget iscsitarget-dkms -y --force-
    yes
scp $PWD/iSCSI/ietd.conf root@$server:/etc/iet/ietd.conf
scp $PWD/iSCSI/iscsitarget root@$server:/etc/default/iscsitarget
ssh root@$server /etc/init.d/iscsitarget restart
ip="`ssh root@$server ifconfig | grep "addr:172\.16\..*\..$" | cut -d : -
    f2 | cut -d ' ' -f1`"

#---------------------------
# Configuration des clients
#---------------------------

scp $HOME/.vimrc root@$client:/root/
scp $PWD/fio/*.fio root@$client:/root/
ssh root@$client apt-get update
ssh root@$client apt-get install fio open-iscsi -y --force-yes
ssh root@$client mkdir /media/partage
ssh root@$client /etc/init.d/open-iscsi restart
ssh root@$client iscsiadm -m discovery --type sendtargets --portal=$ip
ssh root@$client iscsiadm -m node -T asrall.fr -p $ip -l
partition="`ssh root@$client cat /proc/partitions | grep -v sda | grep sd
    | awk '{print $4}'`"
ssh root@$client mkfs.ext4 -q /dev/$partition
ssh root@$client mount /dev/$partition /media/partage

```

```
42 #----------------------------
43 # Affichage des informations
44 #----------------------------
45
46 echo $ip
47 echo "\033[01;33mNoeuds alloues\033[00m"
48 echo "Serveur: \033[01;31m$server\033[00m"
49 echo "Client: \033[01;34m$client\033[00m"
50
51 exit 0
```

RAID0

```
#-------------------------------------
# Script d'installation d'iSCSI
# A utiliser qu'avec 3 noeuds
#-------------------------------------

puts "#------------------------------"
puts "# Deploiement du raid0 de iSCSI"
puts "#------------------------------"

i = 0
serveurs = []
client = ""
ARGV.each do|a|
    if i < 1
        puts "je rentre dans le if"
        client = a
        puts "La valeur du client est: #{client}"
    else
        serveurs << a
    end
    i += 1
end
i -= 1
puts "serveurs: #{serveurs}"
puts "client: #{client}"
#---------------------------
# Configuration du serveur
#---------------------------
ips = {}
serveurs.each do |server|
    system("scp $HOME/.vimrc root@#{server}:/root/")
        system("ssh root@#{server} umount /tmp")
        system("ssh root@#{server} apt-get update")
        system("ssh root@#{server} apt-get install iscsitarget
            iscsitarget-dkms -y --force-yes")
        system("scp $PWD/iSCSI/ietd.conf root@#{server}:/etc/iet/ietd.
            conf")
        system("scp $PWD/iSCSI/iscsitarget root@#{server}:/etc/default/
            iscsitarget")
        system("ssh root@#{server} /etc/init.d/iscsitarget start")
    ip = `ssh root@#{server} ifconfig | grep "addr:172\.16\..*\..$" | cut
        -d : -f2 | cut -d " " -f1`
    ips[server] = ip.to_str
end
#---------------------------
# Configuration du client
#---------------------------
```

```
44 puts "LE CLIENT EST: #{client}"
45 system("scp $HOME/.vimrc root@#{client}:/root/")
46 system("scp $PWD/fio/*.fio root@#{client}:/root/")
47 system("ssh root@#{client} apt-get update")
48 system("echo 'all\n' | ssh root@#{client} apt-get install fio mdadm open-
      iscsi -y --force-yes")
49 system("ssh root@#{client} mkdir /media/partage")
50 system("ssh root@#{client} /etc/init.d/open-iscsi restart")
51 serveurs.each { |server|
52     system("ssh root@#{client} iscsiadm -m discovery --type sendtargets
         --portal=#{ips[server]}")
53     system("ssh root@#{client} iscsiadm -m node -T asrall.fr -l -p #{ips[
         server]}")
54     puts "#{server}: #{ips[server]}"
55 }
56 partition=`ssh root@#{client} cat /proc/partitions | grep -v sda | grep
      sd | awk '{print $4}' | tr '\n' ' ' `
57 p = 0
58 1.upto(i) { |j|
59         partition.insert(p, '/dev/')
60         p += 9
61 }
62 puts "Partition: #{partition}"
63 system("ssh root@#{client} mknod /dev/md0 b 9 0")
64 system("echo 'y\n' | ssh root@#{client} mdadm --create /dev/md0 --level=0
      --raid-devices=#{i} #{partition}")
65 system("ssh root@#{client} mkfs.ext4 /dev/md0")
66 system ("ssh root@#{client} mount /dev/md0 /media/partage")
67 #----------------------------
68 # Affichage des informations
69 #----------------------------
70 ips.each { |ip, serveur| puts "Serveur #{ip} : #{serveur}" }
71 puts "\033[01;33mNoeuds alloues\033[00m"
72 puts "Serveurs: \033[01;31m"+"#{serveurs}"+"\033[00m"
73 puts "Client: \033[01;34m" +"#{client}"+"\033[00m"
74 exit 0
```

RAID50

```
#-------------------------------------
# Script d'installation d'iSCSI
# A utiliser qu'avec 7 noeuds
#-------------------------------------

puts "#------------------------------"
puts "# Deploiement du raid50 de iSCSI"
puts "#------------------------------"

i = 0
serveurs = []
client = ""
ARGV.each do|a|
    if i < 1
        puts "je rentre dans le if"
        client = a
        puts "La valeur du client est: #{client}"
    else
        serveurs << a
    end
    i += 1
end
i -= 1
puts "serveurs: #{serveurs}"
puts "client: #{client}"
#---------------------------
# Configuration du serveur
#---------------------------
ips = {}
serveurs.each do |server|
    system("scp $HOME/.vimrc root@#{server}:/root/")
        system("ssh root@#{server} umount /tmp")
        system("ssh root@#{server} apt-get update")
        system("ssh root@#{server} apt-get install iscsitarget
           iscsitarget-dkms -y --force-yes")
        system("scp $PWD/iSCSI/ietd.conf root@#{server}:/etc/iet/ietd.
           conf")
        system("scp $PWD/iSCSI/iscsitarget root@#{server}:/etc/default/
           iscsitarget")
        system("ssh root@#{server} /etc/init.d/iscsitarget start")
    ip = `ssh root@#{server} ifconfig | grep "addr:172\.16\..*\..$" | cut
        -d : -f2 | cut -d " " -f1`
    ips[server] = ip.to_str
end
#---------------------------
# Configuration du client
#---------------------------
```

```
44 puts "LE CLIENT EST: #{client}"
45 system("scp $HOME/.vimrc root@#{client}:/root/")
46 system("scp $PWD/fio/*.fio root@#{client}:/root/")
47 system("ssh root@#{client} apt-get update")
48 system("echo 'all\n' | ssh root@#{client} apt-get install fio mdadm open-
      iscsi -y --force-yes")
49 system("ssh root@#{client} mkdir /media/partage")
50 system("ssh root@#{client} /etc/init.d/open-iscsi restart")
51 serveurs.each { |server|
52     system("ssh root@#{client} iscsiadm -m discovery --type sendtargets
         --portal=#{ips[server]}")
53     system("ssh root@#{client} iscsiadm -m node -T asrall.fr -l -p #{ips[
         server]}")
54     puts "#{server}: #{ips[server]}"
55 }
56 partition=`ssh root@#{client} cat /proc/partitions | grep -v sda | grep
      sd | awk '{print $4}' | tr '\n' ' ' `
57 p = 0
58 1.upto(i) { |j|
59         partition.insert(p, '/dev/')
60         p += 9
61 }
62 m =  partition.scan(/(\/dev\/sd.)/)
63 part1 ="#{m[0]} #{m[1]} #{m[2]}"
64 part2 ="#{m[3]} #{m[4]} #{m[5]}"
65
66 system("ssh root@#{client} mknod /dev/md0 b 9 0")
67 system("ssh root@#{client} mknod /dev/md1 b 9 0")
68 system("ssh root@#{client} mknod /dev/md50 b 9 0")
69 system("echo 'y\n' | ssh root@#{client} mdadm --create /dev/md0 --level=5
      --raid-devices=3 #{part1}")
70 system("echo 'y\n' | ssh root@#{client} mdadm --create /dev/md1 --level=5
      --raid-devices=3 #{part2}")
71 system("echo 'y\n' | ssh root@#{client} mdadm --create /dev/md50 --level
      =0 --raid-devices=2 /dev/md0 /dev/md1")
72 system("ssh root@#{client} mkfs.ext4 /dev/md50")
73 system ("ssh root@#{client} mount /dev/md50 /media/partage")
74 #----------------------------
75 # Affichage des informations
76 #----------------------------
77 ips.each { |ip, serveur| puts "Serveur #{ip} : #{serveur}" }
78 puts "\033[01;33mNoeuds alloues\033[00m"
79 puts "Serveurs: \033[01;31m"+"#{serveurs}"+"\033[00m"
80 puts "Client: \033[01;34m" +"#{client}"+"\033[00m"
81 exit 0
```

7.1.4 AoE

Simple

```
#!/bin/sh
# Script d'installation de AoE
# A utiliser avec 2 noeuds


echo "#------------------------------"
echo "# Deploiement de AoE"
echo "#------------------------------"

server=$1
client=$2

#---------------------------
# Configuration du serveur
#---------------------------

scp $HOME/.vimrc root@$server:/root/
ssh root@$server apt-get update
ssh root@$server apt-get install vblade -y --force-yes
ssh root@$server modprobe aoe
ssh root@$server umount /tmp
ssh root@$server vblade-persist setup 0 1 eth0 /dev/sda5

#---------------------------
# Configuration du client
#---------------------------

scp $HOME/.vimrc root@$client:/root/
scp $PWD/fio/*.fio root@$client:/root/
ssh root@$client apt-get update
ssh root@$client apt-get install fio aoetools -y --force-yes
ssh root@$client modprobe aoe
ssh root@$client mkdir /media/partage

#---------------------------
# Lancement du service
#---------------------------

ssh root@$server vblade-persist start 0 1
ssh root@$client aoe-discover
ssh root@$client aoe-stat
ssh root@$client mkfs.ext4 /dev/etherd/e0.1
ssh root@$client mount /dev/etherd/e0.1 /media/partage

#---------------------------
# Affichage des informations
```

```
47 #----------------------------
48
49 echo "\033[01;33mNoeuds alloues\033[00m"
50 echo "Serveur: \033[01;31m$server\033[00m"
51 echo "Clients: \033[01;34m$client\033[00m"
52
53 exit 0
```

RAID0

```
#!/bin/sh
# Script d'installation de AoE en RAID0
# A utiliser avec 3 noeuds minimum


echo "#------------------------------"
echo "# Deploiement de AoE (RAID0)"
echo "#------------------------------"

client=$1
shift
servers=$*

#---------------------------
# Configuration du serveur
#---------------------------

i=0
for server in $servers; do
    scp $HOME/.vimrc root@$server:/root/
    ssh root@$server apt-get update
    ssh root@$server apt-get install vblade -y --force-yes
    ssh root@$server modprobe aoe
    ssh root@$server umount /tmp
    ssh root@$server vblade-persist setup $i 1 eth0 /dev/sda5
    i=$(($i+1))
done

#---------------------------
# Configuration du client
#---------------------------

scp $PWD/fio/*.fio root@$client:/root/
scp $HOME/.vimrc root@$client:/root/
ssh root@$client apt-get update
echo "all\n" | ssh root@$client apt-get install fio mdadm aoetools -y --
   force-yes
ssh root@$client modprobe aoe
ssh root@$client mkdir /media/partage

#---------------------------
# Lancement du service
#---------------------------

i=0
devices=""

```

```
47 for server in $servers; do
48     ssh root@$server vblade-persist start $i 1
49     devices="/dev/etherd/e$i.1 $devices"
50     i=$(($i+1))
51 done
52 ssh root@$client aoe-discover
53 ssh root@$client aoe-stat
54 ssh root@$client mknod /dev/md0 b 0 9
55 echo "y\n" | ssh root@$client mdadm --create /dev/md0 --level=0 --raid-
       devices=$i $devices
56 ssh root@$client mkfs.ext4 /dev/md0
57 ssh root@$client mount /dev/md0 /media/partage
58
59 #---------------------------
60 # Affichage des informations
61 #---------------------------
62
63 echo "\033[01;33mNoeuds alloues\033[00m"
64 echo "Serveurs: \033[01;31m$servers\033[00m"
65 echo "Client: \033[01;34m$client\033[00m"
66
67 exit 0
```

RAID50

```
#!/bin/sh
# Script d'installation de AoE en RAID50
# A utiliser avec 6 noeuds

echo "#------------------------------"
echo "# Deploiement de AoE (RAID50)"
echo "#------------------------------"

client=$1
shift
servers=$*

#---------------------------
# Configuration du serveur
#---------------------------

i=0
for server in $servers; do
    scp $HOME/.vimrc root@$server:/root/
    ssh root@$server apt-get update
    ssh root@$server apt-get install vblade -y --force-yes
    ssh root@$server modprobe aoe
    ssh root@$server umount /tmp
    ssh root@$server vblade-persist setup $i 1 eth0 /dev/sda5
    i=$(($i+1))
done

#---------------------------
# Configuration du client
#---------------------------

scp $PWD/fio/*.fio root@$client:/root/
scp $HOME/.vimrc root@$client:/root/
ssh root@$client apt-get update
echo "all\n" | ssh root@$client apt-get install fio mdadm aoetools -y --
   force-yes
ssh root@$client modprobe aoe
ssh root@$client mkdir /media/partage

#---------------------------
# Lancement du service
#---------------------------

i=0
devices_0=""
devices_1=""

```

```
47 for server in $servers; do
48     ssh root@$server vblade-persist start $i 1
49     if [ $i -lt 3 ]; then
50         devices_0="/dev/etherd/e$i.1 $devices_0"
51     else
52         devices_1="/dev/etherd/e$i.1 $devices_1"
53     fi
54     i=$(($i+1))
55 done
56
57 ssh root@$client aoe-discover
58 ssh root@$client aoe-stat
59 ssh root@$client mknod /dev/md0 b 0 9
60 ssh root@$client mknod /dev/md1 b 0 9
61 ssh root@$client mknod /dev/md50 b 0 9
62 echo "y\n" | ssh root@$client mdadm --create /dev/md0 --level=5 --raid-
       devices=3 $devices_0
63 echo "y\n" | ssh root@$client mdadm --create /dev/md1 --level=5 --raid-
       devices=3 $devices_1
64 echo "y\n" | ssh root@$client mdadm --create /dev/md50 --level=0 --raid-
       devices=2 /dev/md0 /dev/md1
65 ssh root@$client mkfs.ext4 /dev/md50
66 ssh root@$client mount /dev/md50 /media/partage
67
68 #---------------------------
69 # Affichage des informations
70 #---------------------------
71
72 echo "\033[01;33mNoeuds alloues\033[00m"
73 echo "Serveurs: \033[01;31m$servers\033[00m"
74 echo "Client: \033[01;34m$client\033[00m"
75
76 exit 0
```

7.1.5 DRDB

Simple

```
#!/bin/sh
# Script d'installation de DRDB
# À utilisé qu'avec 2 noeuds
#-------------------------------------

echo "#------------------------------"
echo "# Deploiement de DRBD"
echo "#------------------------------"

primary=$1
secondary=$2
ip_primary="`ssh root@$primary ifconfig | grep "addr:172\.16\..*\..$" | cut -d : -f2 | cut -d ' ' -f1`"
ip_secondary="`ssh root@$secondary ifconfig | grep "addr:172\.16\..*\..$" | cut -d : -f2 | cut -d ' ' -f1`"

#---------------------------
# Configuration du primaire
#---------------------------

scp $HOME/.vimrc root@$primary:/root/
scp $PWD/fio/*.fio root@$primary:/root/
ssh root@$primary apt-get update
ssh root@$primary apt-get install fio drbd8-utils -y --force-yes
scp $PWD/drbd/r0.res root@$primary:/etc/drbd.d/
ssh root@$primary sed -r "s/PRIMARY/$primary/g" -i /etc/drbd.d/r0.res
ssh root@$primary sed -r "s/SECONDARY/$secondary/g" -i /etc/drbd.d/r0.res
ssh root@$primary sed -r "s/IPP/$ip_primary/g" -i /etc/drbd.d/r0.res
ssh root@$primary sed -r "s/IPS/$ip_secondary/g" -i /etc/drbd.d/r0.res
ssh root@$primary umount /tmp
ssh root@$primary mkfs.ext4 /dev/sda5
ssh root@$primary e2fsck -f /dev/sda5
ssh root@$primary resize2fs /dev/sda5 100G
echo "yes\n" |ssh root@$primary drbdadm create-md r0
ssh root@$primary mkdir /media/partage
ssh root@$primary mount /dev/drbd0 /media/partage

#---------------------------
# Configuration du secondaire
#---------------------------

scp $HOME/.vimrc root@$secondary:/root/
ssh root@$secondary apt-get update
ssh root@$secondary apt-get install drbd8-utils -y --force-yes
scp $PWD/drbd/r0.res root@$secondary:/etc/drbd.d/
ssh root@$secondary sed -r "s/PRIMARY/$primary/g" -i /etc/drbd.d/r0.res
```

```
45 ssh root@$secondary sed -r "s/SECONDARY/$secondary/g" -i /etc/drbd.d/r0.
      res
46 ssh root@$secondary sed -r "s/IPP/$ip_primary/g" -i /etc/drbd.d/r0.res
47 ssh root@$secondary sed -r "s/IPS/$ip_secondary/g" -i /etc/drbd.d/r0.res
48 ssh root@$secondary umount /tmp
49 ssh root@$secondary mkfs.ext4 /dev/sda5
50 ssh root@$secondary e2fsck -f /dev/sda5
51 ssh root@$secondary resize2fs /dev/sda5 100G
52 echo "yes\n" |ssh root@$secondary drbdadm create-md r0
53
54
55 #----------------------------
56 # Lancement de DRBD
57 #----------------------------
58
59 ssh root@$primary modprobe drbd
60 ssh root@$secondary modprobe drbd
61 ssh root@$primary drbdadm up r0
62 ssh root@$secondary drbdadm up r0
63 ssh root@$primary drbdadm -- --overwrite-data-of-peer primary r0
64
65 #----------------------------
66 # Affichage des informations
67 #----------------------------
68
69 echo "\033[01;33mNoeuds alloues\033[00m"
70 echo "Primaire: \033[01;31m$primary\033[00m"
71 echo "Secondaire: \033[01;34m$secondary\033[00m"
72
73 exit 0
```

7.1.6 Scripts de réassemblage

Nous avons fait des scripts d'assemblage de RAID pour chaque technologie. Étant donné que ces scripts fonctionnent de la même manière, voici un script d'assemblage pour la technologie NBD et iSCSI (pour le RAID 50).

Assemblage RAID simple

```
#!/bin/sh
# Script pour remonter un raid  de NBD (RAID0,1,5,6,10)
# A utiliser qu'avec 3 noeuds ou plus
# Le premier argument doit etre le client suivi des serveurs
#-------------------------------------

echo "#------------------------------"
echo "# Remontage de NBD (RAID0,1,5,6,10)"
echo "#------------------------------"

client=$1
shift
servers=$*

i=0
devices=""

for server in $servers; do
    ssh root@$client nbd-client $server 2000 /dev/nbd$i
    devices="/dev/nbd$i $devices"
    i=$(($i+1))
done

ssh root@$client mknod /dev/md0 b 9 0
ssh root@$client mdadm --assemble /dev/md0 $devices
ssh root@$client mount /dev/md0 /media/partage

exit 0
```

Assemblage RAID 50

```
# Script pour remonter un raid  de iSCSI pour le raid 50
# A utiliser qu'avec 7 noeuds ou plus
# Le premier argument doit etre le client suivi des serveurs
#-------------------------------------

puts "#-------------------------------------"
puts "# Remontage de iSCSI (RAID0,1,5,6,10)"
puts "#-------------------------------------"

i = 0
serveurs = []
client = ""
ARGV.each do|a|
        if i < 1
                puts "je rentre dans le if"
                client = a
                puts "La valeur du client est: #{client}"
        else
                serveurs << a
        end
        i += 1
end
i -= 1

serveurs.each { |server|
        ip = `ssh root@#{server} ifconfig | grep "addr:172\.16\..*\..$" |
             cut -d : -f2 | cut -d " " -f1`
        system("ssh root@#{client} iscsiadm -m discovery --type
           sendtargets --portal=#{ip}")
        system("ssh root@#{client} iscsiadm -m node -T asrall.fr -l -p #{
           ip}")
        puts "#{server}: #{ip}"
}
partition=`ssh root@#{client} cat /proc/partitions | grep -v sda | grep
    sd | awk '{print $4}' | tr '\n' ' ' `

p = 0
1.upto(i) { |j|
        partition.insert(p, '/dev/')
        p += 9
}
m =  partition.scan(/(\/dev\/sd.)/)
part1 ="#{m[0]} #{m[1]} #{m[2]}"
part2 ="#{m[3]} #{m[4]} #{m[5]}"
system("ssh root@#{client} mknod /dev/md0 b 9 0")
system("ssh root@#{client} mknod /dev/md1 b 9 0")
system("ssh root@#{client} mknod /dev/md50 b 9 0")
```

```
44 system("ssh root@#{client} mdadm --assemble /dev/md0 #{part1}")
45 system("ssh root@#{client} mdadm --assemble /dev/md1 #{part2}")
46 system("ssh root@#{client} mdadm --assemble /dev/md50 /dev/md0 /dev/md1")
47 system("ssh root@#{client} mount /dev/md50 /media/partage")
48
49 exit 0
```