TIW - Cloud computing

Stockage

Fabien Rico (fabien.rico@univ-lyon1.fr)
Jean Patrick Gelas

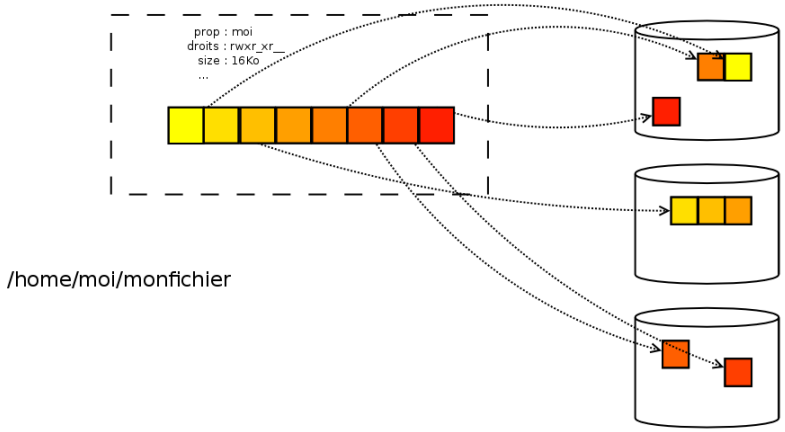
Univ. Claude Bernard Lyon 1

séance 4

Système de Fichiers

Le rôle d'un système de fichiers est de socker et retrouver des données.

- Les données sont lues par *bloc* c'est à dire par morceau de taille fixée par le système, le matériel, ...
- Les données peuvent être éparpillées
  - sur un même disque,
  - sur des disques différents,
  - sur des serveurs différents,
  - ...



- Le point d'entrée d'un fichier
  - est le moyen de retrouver les différents blocs sur leur support ;
  - contient en plus des informations sur le fichier les *méta-données* ;
  - est associé à un identifiant ou à un nom dans un *espace de nomage*.

Le bloc

C'est l'unité de base de l'écriture ou la lecture.

- On ne peut pas écrire ou lire moins d'un bloc.
- Importance du cache mémoire.
- Problème de la cohérence du système de fichiers :
  - après un arret brutal ;
  - après des accès concurents ;
  - après une coupure réseau dans un système réparti.
- Duplication de blocs.
- Efficacité des accès en fonction de la répartition des blocs.
- Lecture en parallèle dans plusieurs disques ?

Répartition des blocs

Pour être efficace, il faut que *le système accède facilement aux blocs.*

- Dépend de la façon dont la lecture est faite :
  - Lecture séquentielle dans un même disque
    - ↦ blocs successifs
    - ↦ défragmentation.
  - Lecture en parrallèle dans plusieurs disques
    - ↦ maximisation du parallélisme
    - ↦ les blocs doivent être sur des disques séparés.
  - Lecture d'une base de données
    - ↦ ? ? ?
- Et si on ajoute le réseau ?

Qu'est-ce qu'un fichier ?

- Les blocs de données du fichier.
- Une structure de données permettant de les retrouver
  - appelée *inode* dans certains cas ;
  - permet de reconstituer le fichier ;
  - contient des informations sur le fichier.
- Des *méta-données*
  - propriétaire, groupe...
  - droit, ACL...
  - dates de modification, de lecture, ...
  - ...
  - champs clef/valeur
- Un nom et une position dans le système de fichiers
  - des fichiers spéciaux, les répertoires ;
  - une structure arborescente ;
  - *... cette partie n'est pas obligatoire.*

Structure de système de fichiers

Traditionnellement un système de fichiers est d'abord vu comme un arbre où chaque fichier est dans un chemin de racine$\mapsto$ répertoire $\mapsto$ sous répertoire...
Mais cela a tendance à changer :

- Introduction d'arborescences supplémentaires basées sur les versions, *sous volumes* (vois plus loin).
- Possibilité de fusionner des sous arbres : *union mount* (voir plus loin).
- Organisation semblable à une base de données : *stockage objet*.

Stockage objet

- Un fichier peut être vu comme un élément associé à un identifiant.
- L'ensemble de fichiers est plat (pas de hiérarchie ou de chemin) comme une table de base de données
- Les métadonnées peuvent servir à faire des requêtes.
- Elles n'ont pas de rôle prédéfini.

Exemple (Amazon S3)

- Les fichiers sont récupérés à partir de leur identifiant via une API web.
- Les métadonnées sont sous forme clef/valeur.
- Il est possible de faire des requêtes en fonction de ces métadonnées
- Organisation dans des seaux (*buckets*)
- Configuration via ces seaux.
- Différents niveaux de stockage plus ou moins rapides (et coûteux).

Problématiques

- Les capacité des disques ont beaucoup augmenté :
  - qqs 10zaines de Go en 2000 ;
  - qqs To en 2015.
- La vitesse est limitée par la mécanique < 15000 tours/s.
  - problème pour les opérations longues (défragmentation, test de cohérence, sauvegarde)
- Le taux d'erreur est resté le même :

Si un disque a 1 chance sur 10 d'être défectueux au bout d'un an, combien de chance d'avoir une erreur avec un serveur de 10 disques ?

Opération à chaud

Les systèmes fichiers sont capables d'effectuer certaines tâches à chaud (alors qu'ils sont utilisés).

- défragmentation ;
- vérification de la cohérence ;
- reconstruction (voir RAID) ;
- sauvegarde (voir snapshot).

Cela permet de ne pas stopper le système pour faire ces opérations et redémarrer plus rapidement.
Rappel, ces opérations sont forcement lentes !

Instantané (*Snapshot*)

Définition (Instantané)

C'est l'action de sauvegarder un système de fichiers tel qu'il est à un instant donné. Cela est fait malgré les changements sur le système durant la sauvegarde.

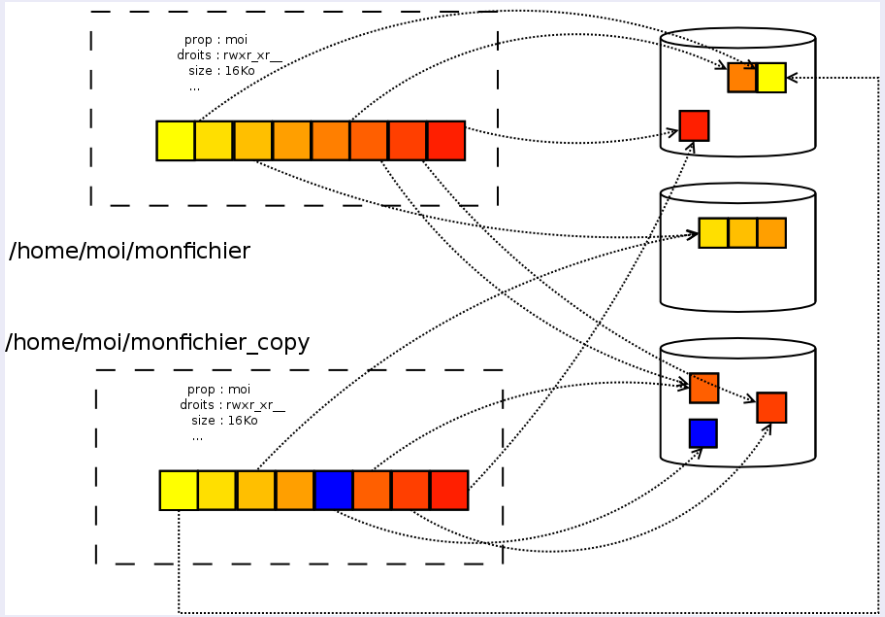
- Nécessaire pour faire une sauvegarde cohérente.
- Utile pour conserver différentes versions d'un objet (disque de VM par ex.)
- Besoin de conserver les blocs modifiés et leur version originale.
- Comment fait-on ?

Copy on Write (COW)

Définition (Copy on Write)

C'est le fait de conserver en commun les parties non modifiées lorsqu'on fait une copie.

- Juste après la copie, les 2 fichiers utilisent les mêmes blocs de données. Seules les métadonnées et les *inodes* sont copiées.



- S'il y a modification du fichier on dupplique le(s) bloc(s) concerné(s) uniquement.

Copy on Write (suite)

On a donc :

- deux fichiers qui semblent différents mais occupent beaucoup de blocs en commun.

+ un gain de place ;
+ un gain de temps à la copie ;
+ une répartition dans le temps de ce qui coûte le plus cher (la copie des blocs de données) ;
- un surcout pour les entrées/sorties après la copie.

Certains systèmes gèrent naturellement le COW (données réparties),
Certaines utilisations en ont besoin (Virtualisation).

Exemple de BTRFS

Exemple (COW BTRFS)

Les données sont modifiées de la même manière que dans une base de données transactionnelle :

- Si je modifie des blocs.
- Les nouvelles valeurs sont recopiées ailleurs et les anciens blocs sont conservés.
- Lorsque l'écriture est terminée, les métadonnées sont modifiées pour pointer sur les nouveaux blocs.

- On peut revenir en arrière en cas de problème.
- On peut conserver les anciens blocs.
- On peut garder et faire évoluer différentes versions (les *sous volumes*).
- Cela ne cause pas d'écriture supplémentaire.
- Cela cause de la fragmentation.

Peut-on allez plus loin ?

Example

- Sur un serveur de virtualisation vous déployez 2 machines à partir de la même base.
- Puis vous les mettez à jour.
- Les 2 machines sont identiques, mais les blocs modifiés par la mise à jour sont séparés car ils ont été modifiés.

Peut-on remettre en commun les blocs identiques ?

- Oui c'est la *déduplication*.
- Comment ? (voir plus tard).
- Pourquoi ?
  - sauvegardes ;
  - virtualisation ;
  - ...

Fichiers creux

Il est possible de ne pas affecter de blocs physiques à tous les blocs logiques d'un fichier.

- Certains fichiers sont *creux* (ont de grandes parties vides ou remplie de 0) :
  - Un fichier de machine virtuelle ;
  - Un fichier en cours de téléchargement.
- Si on n'attribue les blocs uniquement aux parties non vides :
  - on économise de la place ;
  - on répartie mieux les données utiles ;
  - on diminue les temps de copie, stockage ...

Union mount

On peut créer des systèmes de fichiers à partir du mélange de plusieurs autres systèmes.

- Soit au niveau des blocs de données comme BTRFS
- Soit au niveau des fichiers *Union mount*
  - AUFS et UNIONFS sous linux
  - LIBRARY sous windows

Définition (*union mount*)

Le *montage en union* est un moyen de créer un répertoire à partir de l'union de différents répertoires.

- Les fichiers qui ne sont présents que dans l'un des systèmes sont présents dans l'union.
- Pour ceux qui sont présents dans les 2, c'est le plus haut qui est dans l'union.
- Les modifications sont écrites dans le plus haut modifiable.

Par exemple aufs (Docker, salle TP réseau), funionmount, ...

Volume logique

De plus en plus souvent il y a une couche d'abstraction entre le système de fichiers et le(s) disque(s). C'est le *volume logique*.

Définition (Volume logique)

C'est le fait de présenter au système un disque virtuel dont les données seront réparties et/ou dupliquées sur les disques disponibles.

Cela peut être fait

- par le matériel (cartes RAID) ;
- par le logiciel (LVM).

Cela permet :

- d'agréger des disques ;
- d'utiliser la redondance (RAID 1, 5, 6...) ;
- d'utiliser le parallélisme (RAID 0) ;
- de mettre en place les *snapshot*, les opérations à chaud...

Le même principe peut être adapté à des disques répartis sur plusieurs ordinateurs différents, c'est un *système de fichier distribué*.

Définition (Système de fichiers distribués)

C'est un système de fichiers dont les données sont réparties sur plusieurs serveurs.

Il en existe un grand nombre selon

- les besoins ;
- le matériel utilisé.

Pourquoi ?

Avantage

- Diminuer les coûts : un grand nombre de petits disques.
- Répartir la charge : le temps d'accès aux données sur un disque évolue peu (vitesse de rotation).
- Évolution : possibilité d'ajouter un serveur à l'ensemble pour augmenter la place.

Inconvénient :

- Pannes : on multiplie les matériels donc les chances de défaillance.
- Difficulté d'administration.
- Risque de goulot d'étranglement.

Besoins

- Disponibilité (en cas de défaillance d'un serveur ou du réseau)
- Intégrité (en cas de problème matériel)
- Cohérence (écritures concurrentes, cache)
- Efficacité d'accès (débit et latence)

Pour avoir un fichier il faut :

- Un nom, des droits d'accès, connaître le lieu de stockage
  - ⇒ espace de nom unique ;
  - ⇒ système de gestion des utilisateurs ;
  - ⇒ serveur(s) de *métadonnées*.
- Un contenu :
  - ⇒ serveur(s) de données
- Un accès
  - utilise-t'on service centralisé ?
  - utilise-t'on un cache ?

Duplications des données

- Les données peuvent être dupliquées
  - pour maintenir l'intégrité (duplication du contenu)
  - pour améliorer l'accès (contenu et métadonnés)
  - pour la disponibilité (métadonnées, pas de points de défaillance)
- Il y a toujours un système de cache
  - sur un serveur « plus proche » ;
  - sur le client ;
  - en mémoire (par exemple pour les métadonnées).

S'il y a duplication, il faut résoudre les problèmes de cohérence.

Modèles

Les propriétés du système de fichiers seront différentes selon la façon dont on gère les données :

- Que se passe-t'il si on découpe les fichiers en blocks (ou *chunk*) pour les répartir sur plusieurs serveurs ?
- Que se passe-t'il si on duplique les fichiers (ou les chunk) sur plusieurs serveurs ?

Modèles

Selon la gestion des métadonnées, il y a différents modèles.

- Semi-centralisé : un serveur de métadonnées, des serveurs de données
  - problème en cas de défaillance du serveur de métadonnées ;
  - disponibilité moyenne.
- Décentralisé : les données et métadonnées sont réparties
  - plus complexe ;
  - meilleure disponibilité.

Exemple : Moosefs

- Compatible POSIX
- Semi-centralisé
  - le serveur de métadonnées stocke tout en mémoire pour des raisons d'efficacité ;
  - utilisation d'un log pour reconstruire le serveur ;
  - possibilité d'utiliser des serveurs de secours.
- Les fichiers sont découpés et dupliqués automatiquement.
- Réparti la charge en fonction de la taille des disques.

Exemple : Lustre

- Première version en 2003, racheté par SUN puis Oracle
- Compatible POSIX
- Semi centralisé,
  - métadonnées sur un seul système de fichiers
  - propose d'utiliser plusieurs serveurs pour assurer la disponibilité
  - nécessité de protéger les métadonnées (RAID)
- Les fichiers sont découpés et dupliqués automatiquement de plus il y a possibilité d'installer un volume logique et un RAID logiciel.
- Système présent dans de nombreux super-calculateurs.

Exemple : Google File System

- Développé par Google pour son utilisation
- Propriétaire
- Semi centralisé,
  - le serveur de métadonnées est dupliqué.
- Les fichiers sont découpés et dupliqués automatiquement.
- Définition optimisée pour l'ajout dans les fichiers.

Exemple : General Parallel File System

- Développé par IBM, commencé en 1993 et dernière version en 2010.
- Compatible POSIX.
- Décentralisé
  - les métadonées sont réparties ;
  - pas de point de défaillance.
- Les fichiers sont découpés en petits fichiers mais pas forcement dupliqués
  - rapidité d'accès ;
  - pour l'intégrité, il est nécessaire d'utiliser de protéger les systèmes de fichiers de base (RAID).

Exemple : HDFS (Hadoop)

- Développé par « Apache Software Foundation » 1ère version en 2011, dernière en juillet 2015.
- Non compatible POSIX (les besoins sont différents).
- Semi-décentralisé
  - les métadonnées sont gérées par un *NameNode* ;
  - les données par plusieurs *datanode ;*
  - *possibilité d'avoir un* NameNode *secondaire qui recopie régulièrement le principal.*
- Les fichiers sont découpés et recopiés sur plusieurs noeuds (en tenant compte de la proximité)
  - rapidité d'accès ;
  - disponibilité en cas de panne ;
  - utilisation de la localisation pour l'affectation de tâches map/reduce ;
  - orienté vers la lecture ou création de fichier (immutable), ce n'est pas efficace pour la modification concurrente.

Exemple : CEPH

- Développé par un consortium (Canonical, CERN, Cisco, Fujitsu, Intel, RedHat, ScanDisk ans SUSE).
- 2 couches :
  - Les clients obtiennent un *lun* : disque virtuel formé de blocs RBD (*RADOS Block Device*).
  - Les blocs sont des objets dans un stockage objets réparti (RADOS = *Reliable Autonomic Distributed Object Store*).
- Décentralisé :
  - Des moniteurs pour vérifier les noeuds de stockage, décider et informer de la localisation des blocs.
  - Des serveurs dupliqués pour le reste des métadonnées.
  - Les blocs RDB sont dupliqués au moins une fois, un même LUN n'est jamais dans un seul serveur de stockage.
- Utilisé dans notre plateforme OPENSTACK pour les volumes :
  - instantané de volumes ;
  - créations rapide de VM via les volumes grâce au copy-on-write.

Architecture courante

- Les systèmes de fichiers distribués
  - sont souvent difficiles à déployer ;
  - nécessitent une interconnexion efficace entre les différents serveurs.
- En général ils sont réalisés sur un réseau dédié
  - réseaux haut débit (jumbo frame, myrinet, infiniband, ...) ;
  - sécurité de l'accès aux données ;
  - évite les perturbations.
- Les clients sont des machines ou logiciels qui utilisent ce système pour stocker leur données.
  - Par exemple, des serveurs de base données qui stockent leur table.
  - Par exemple, des serveurs de virtualisation qui stockent les disques de leur VM.

Partage de fichiers

Définition

C'est le fait, pour un serveur, de donner accès à ses fichiers ou à ses disques à une ou plusieurs autres machines.

- Une fois mis en place l'accès est transparent pour les applications :
  - disque partagé (compte informatique) ;
  - baie de stockage ;
  - accès grande distance (DROPBOX, OWNCLOUD, ...).
- Centraliser le stockage :
  - pour améliorer sa gestion ;
  - pour créer des postes de travail interchangeables ;
  - pour synchroniser plusieurs applications qui tournent en parallèle (*load balancing*).

Un peu de jargon

- NAS (Network Attached Storage) : matériel spécialisé dans le partage de fichiers via les protocoles courants (ftp, nfs, smb, ...). On peut utiliser une machine avec un système dédié :
  - Freenas
  - Windows Storage Serveur
- SAN (Storage Area Network) : réseau de disques, unités partageant l'accès à leurs disques durs
  - accès en mode block ;
  - les clients obtiennent un disque virtuel (LUN) ;
  - les clients gèrent le système de fichiers.

Cas du *load balancing*

Lorsque vous voulez répartir la charge d'une application sur plusieurs serveurs, il faut un stockage commun.

- TP docker : deux instances de l'application web mais une seule base.
- Serveurs de virtualisation : le déplacement a chaud de VM n'est possible que si les disques des VM sont sur un support partagé.
- Base de donnée en cluster oracle :
  - Une instance de base de données est un ensemble de processus qui gère les client pour une base particulière.
  - Pour que plusieurs instances soient associées à la même base (cluster), il faut au minimum que les fichiers de données et de reprise soient partagés.
  - Lorsque un client se connecte, une même session est toujours gérée par une même instance.

Notion de cache

Un partage implique un accès au données via le réseau, ce qui est lent. Il faut donc un cache sur le client.

- taille des données
  - fixe (les fichiers sont découpés en blocs lus en un seul morceau et stockés localement)
  - variable (cache de fichiers, mode déconnecté)
- propagation des modifications
  - immédiate *write through*
  - différée *write back*
- algorithme de chargement
  - à la demande
  - pré-chargement (ex streaming)
- algorithme de remplacement
  - FIFO, LRU, LFU, ...

Cohérence des données

S'il y a partage, il peut y avoir modification concurrente ou modification et lecture.

- Visibilité immédiate
  - ce qui est vu est la dernière version
  - difficile de définir « dernière »
  - ex : système lecteur/rédacteur
- Session
  - En début de session récupération d'une copie sur laquelle on travaille
  - Les modifications ne sont visibles qu'à la session suivante.

Network File System (NFS)

- Présenté par Sun en 1985.
- permet à ses stations sans disque d'accéder à un système de gestion de fichiers distant s(RFC 1094).
- Utilise les appels de procédures distantes Sun-RPC (qui sont issues des travaux sur NFS).
- A priori, le client et le serveur NFS devraient être des processus utilisateur s'exécutant au-dessus de RPC/XDR/UDP/IP. Mais en fait, le client et le serveur sont des modules du noyaux.
- La dernière version (version 4) est totalement différente (et incompatible) avec les précédentes. On parle donc séparément de NFSv3 (ancienne version) et NFSv4 (nouvelle).

NFSv3 : en pratique ...

Exemple (Client NFSv3)

```
root@192.168.69.1# rpcinfo -u 192.168.69.2 showmount
program 100005 version 3 ready and waiting
root@192.168.69.1# rpcinfo -u 192.168.69.2 nfs
program 100003 version 3 ready and waiting
root@192.168.69.1# showmount -a 192.168.69.2
All mount points on 192.168.69.2:
192.168.69.1:/home
root@192.168.69.1# showmount -d 192.168.69.2
Exported directories on 192.168.69.2:
/home
root@192.168.69.1# showmount -e 192.168.69.2
Export list for 192.168.69.2:
/home 192.168.69.0/255.255.255.0
```

Exemple (Serveur NFSv3)

```
root@192.168.69.2# showmount
Hosts on 192.168.69.2:
192.168.69.1
```

NFSv4

- Largement inspiré de AFS et CIFS.
- N'utilise plus le protocole UDP.
- Délégation de gestion du fichier sur le client (cache)
- Meilleure sécurité :
  - authentification des utilisateurs du client,
  - chiffrage des échanges.
- Amélioration de la montée en charge.

Gestion de cache

Pour permettre l'utilisation de caches locaux, le serveur doit déléguer la gestion d'un fichier à l'un des clients :

- soit une délégation pour la lecture (signifiant que le client peut lire le fichier sans demander au serveur s'il a été modifié)
- soit une délégation pour l'écriture (signifiant que le serveur peut écrire sur le fichier sans prévenir le serveur)

Questions :

- Quelles sont les contraintes sur ces délégations ?
- En considérant qu'un client peut se déconnecter brutalement, quelles sont les sécurités à appliquer aux délégations ?
- Quelles choix de l'administrateur peuvent avoir un effet sur les performances du système ?

Différence de configuration avec NFS3

- Existence d'un *pseudo-filesystem*.
  - racine des fichiers exportés (attribut fsid =0) ;
  - tous les répertoires exportés doivent être des sous-répertoires de la racine ;
  - il est nécessaire d'ajouter un répertoire que l'on veut exporter à cette racine (commande mount −−bind) ;
  - le client ne voit que le *pseudo filesystem*
- Possibilité d'authentification kerberos et de chiffrage, sur le serveur on spécifie
  - krb5 : vérification de l'identité de celui qui accède ;
  - krb5i : vérification de l'intégrité des données (non modifiées) ;
  - krb5p : protection des données (chiffrage).

NFSv4 : en pratique

Exemple (Client NFSv4)

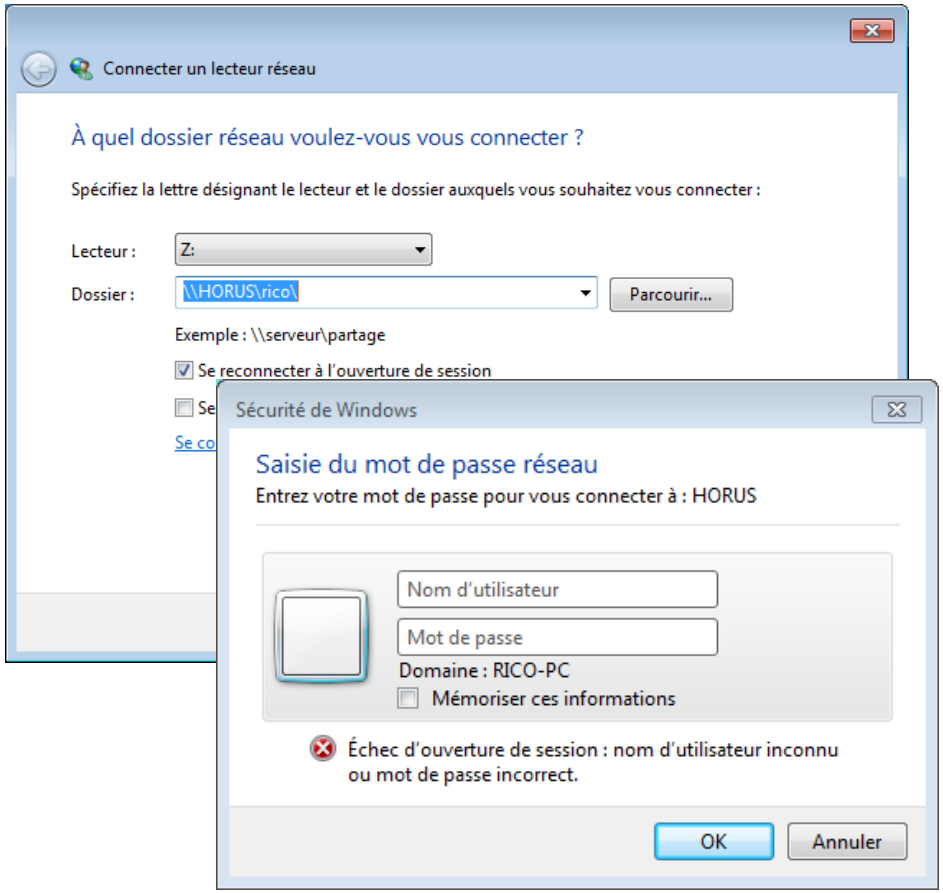
```
root@192.168.69.1# tail -1 /etc/fstab
192.168.69.2:/home  /nfshome  nfs4  defaults,noauto 0 0
root@192.168.69.1# mkdir /nfshome
root@192.168.69.1# mount  /nfshome
root@192.168.69.1# ls -l /nfshome
drwxr-xr-x    2 1003     5000         1024 fev 16 13:32 olivier
root@192.168.69.1# df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda2              3898108   2766592    930304  75%    /
192.168.69.2:/home    16128636   1493328  13815996  10%    /nfshome
```

Exemple (Serveur NFSv4)

```
root@192.168.69.2# mkdir -p /nfs4/home
root@192.168.69.2# mount --bind /home /nfs4/home
root@192.168.69.2# cat /etc/exports
/nfs4/ 192.168.69.0/255.255.255.0(rw,fsid=0,insecure,no_subtree_check)
/nfs4/home 192.168.69.0/255.255.255.0(rw,insecure,no_subtree_check)
```

SMB : Server Message Block

Protocole de Microsoft et Intel permettant le partage de ressources (disques, imprimantes. . .) à travers un réseau (1987). Il a connu plusieurs évolutions (et différents noms SMB, CIFS et SMB2)



- On peut se connecter à un répertoire partagé par un serveur distant.
- On peut lui attribuer une lettre (le *monter* ?)
- L'utilisateur doit s'identifier auprès du serveur.

SMB : Groupe de machines

- Notion de groupe de travail
  - Ensemble de machines qui partagent leurs données avec SMB
  - Un client qui essaye d'accéder à un partage doit s'authentifier
    - ★ par un mot de passe unique associé au partage (mode partage) ;
    - ★ par un couple (utilisateur/passwd)
- Notion de domaine
  - un ensemble d'utilisateurs (avec nom et mot de passe) et de serveurs (avec des droits d'accès)
  - un *primary domain controler (PDC)* contient la base de données des utilisateurs et de leurs mots de passe. Il peut être répliqué par des *domain controlers (DC)*.
  - Les autres serveurs délèguent les authentifications aux DCs.
- Active Directory : domaine basé sur un annuaire, voir cours d'administration windows.

Conclusion

- De votre point de vu, il est toujours possible de créer des machines ou des systèmes de fichiers de taille énorme.
- Mais il y a toujours une limite matériel.
- Dans un système de grande taille, le réseau doit entrer en ligne de compte.
- Cela justifie l'apparition de modèles de calcul comme Spark.