NFS – Network File System

RSX 102
Anne WEI
CNAM Paris

1

Bibliographie

- Sun Microsystem «NFS: Network File System Protocol Specification », mars 1989, RFC 1094, IETF
- B. Callaghan and al «NFS Version 3 Protocol Specification », Juin 1995, RFC 1813, IETF
- S. Shepler and al « Network File System (NFS) version 4 Protocol », 2003, RFC 3530 (qui remplace RFC 3010 de 2000), IETF
- Gérard Florin, support de cours RSX102 « Technologie des applications Client-serveur », 2000

2

Plan

1. Introduction
2. Systèmes de fichiers
3. Protocole NFS
4. Configuration NFS
5. Conclusion

3

Introduction (1)

- NFS *(Network File System)* créé par Sun Microsystem en 1984 a pour but de partager les fichiers via un réseau
- NFS fonctionne au dessus de XDR et RPC

	Applications TCP/IP directes				Applications pile SUN/OS
		EXEMPLES			
7. Application	**DNS: Domain Name System**	**SMTP: Simple Mail Transfer Protocol**	**HTTP: Hyper Text Transfer Protocol**	**FTP: File Transfer Protocol**	**NFS: "Network File System"**
6. Présentation					**XDR:"External Data Representation"**
5. Session					**RPC:"Remote Procedure Call"**
4. Transport	**TCP: Transmission Control Protocol (connecté)** **UDP: User Datagram Protocol (non connecté)**				
3. Réseau	**IP: Internet Protocol**				
2. Liaison	**Encapsulation IP (sur LAN ou liaisons SLIP,PPP)**				
1. Physique	**Pratiquement tout support de transmission** **Réseaux Publics**	**Lignes spécialisées Point à Point**		**Réseaux Locaux**	**Réseau téléphonique RNIS, ATM**

Introduction (2)

- NFS permet de partager les fichiers à distance d'une manière transparente comme le partage des fichiers dans le stockage local
- NFS est un protocole se basant sur le mode synchrone (ou asynchrone) – client/serveur. C'est-à-dire, le client lance une requête et le serveur répond par un message avec un statut et les données demandées.
- De vue client, NFS est *un protocole d'accès* à des fichiers distants.
- De vue serveur, NFS est *un service* de mise à disposition de fichiers
- UDP a été défini comme le protocole de transport dans la norme NFS2 TCP a été rajouté comme le protocole de transport dans la norme NFS3

Historique

- Sun Microsystem a crée NFS en 1984. NFS est normalisé par l'IETF comme NFS2 en 1989 (RFC 1094). NFS2 est un protocole *sans état* (serveur ne maintient pas d'info concernant l'exécution de requête) et *synchrone*. La taille maximum de fichiers à lire est de 2Go.
- NFS3 soutenu toujours par Sun Microsystem est normalisé par l'IETF en 1995 (RFC 1813). Cette technologie permet aux fichiers de dépasser la talle de 2Go ; les écritures asynchrones sur le serveur sont possibles ; les écritures sur le client sont en mémoire cache ; le contrôle d'accès est effectué avant les manipulations sur les fichiers
- NFS4 normalisé par RFC 3530 en 2003 (qui remplace RFC 3010 de 2000). Les contributions principales de NFS4 consistent à étendre le mode de connexion et l'accès multiple (de LAN à WAN)
- En mars 2012, une nouvelle version NF4 est en cours de discussion ….

Plan

1. Introduction
2. Systèmes de fichiers
3. Protocole NFS
4. Configuration NFS
5. Conclusion

7

Introduction

- Systèmes de fichiers consistent à stocker, à organiser, à traiter et à accéder aux fichiers locaux ou à distant. Les aspects à traiter sont :
- *Nom de fichier* : un nom de fichier est associé à un répertoire. L'organisation de répertoires est généralement en arbre dans un système. Par exemple, un nom de fichier accessible par Web (voir NFSv4).

```
http://www.cnam.fr/ABU/principal/ABU.u2.html
```

protocole serveur ◄——————► accès au fichier

- *descripteurs de fichiers* : la talle de fichier, la date/heure de création, la date/heure de dernière modification, le type de stockage (en bloque, en caractère…), l'Identifiant et le droit d'accès. Un profil de fichier peut être associé à un fichier d'extension.
- *Accès sécurisé* : une liste de contrôle d'accès ou le droit d'accès

8

Introduction

- Systèmes de fichiers consistent à stocker, à organiser, à traiter et à accéder aux fichiers locaux ou à distant. Les aspects à traiter sont:
- *Nom de fichier*: un nom de fichier est associé à un répertoire.
  L'organisation de répertoires est généralement en arbre dans un système.
  Par exemple, un nom de fichier accessible par Web (voir NFSv4).

```
http://www.cnam.fr/ABU/principal/ABU.u2.html
protocole  serveur  <------------------------>
                          accès au fichier
```

- *descripteurs de fichiers* : la taille de fichier, la date/heure de création, la date/heure de dernière modification, le type de stockage (en bloque, en caractère…), l'identifiant et le droit d'accès. Un profil de fichier peut être associé à un fichier d'extension.
- *Accès sécurisé* : une liste de contrôle d'accès ou le droit d'accès

9

Implantation de fichiers (1)

- Accès séquentiel : accès au fichier depuis le début (allocation contigië)
- Accès direct par adressage dispersé en fonction d'une clé (*Figure 1*)
- Accès direct par index sur une clé (*Figure 2*). Dans ce cas, on peut utiliser un index primaire et des index secondaires

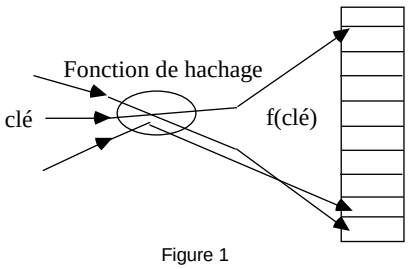


Figure 1

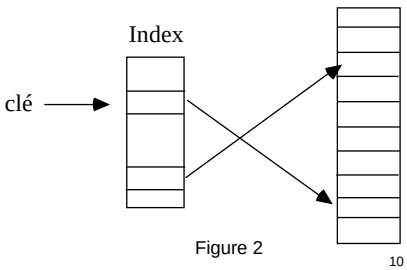
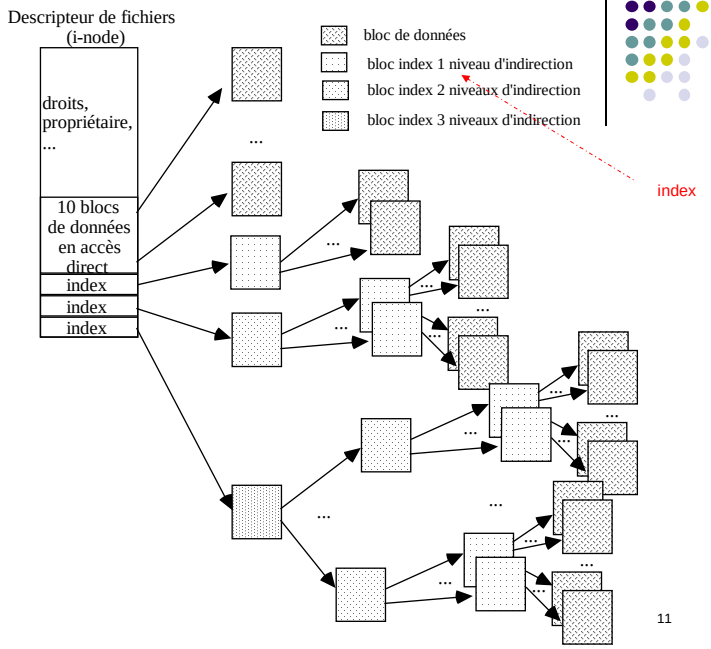


Figure 2

10

- fichiers non structurés, de type Unix ou MS-DOS, les plus répandus



11

Implantation de fichiers (1)

- Les fichiers contiennent des *données, stockées sur un support physique* (disque)
- Ces données sont accessibles à travers des *noms* de fichiers
- les données sont repérées de façon interne non pas par un nom, mais par un numéro sous Unix, le ***i-noeud***, indépendant de l'adresse réelle des données sur le disque
- Cet *i-noeud* assure une interface entre l'utilisateur et le système physique de fichiers (voir des détails plus tard).

12

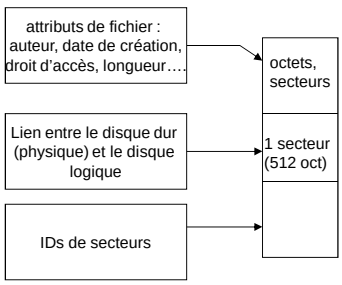
Implantation de fichiers (2)

- Un système de fichiers doit
  - associer à un numéro de périphérique logique
  - contenir un bloc d'information de démarrage (Boot bloc)
  - contenir les information de gestion (super bloc)
  - contient une suite d'i-noeuds sous Unix, descripteurs de fichiers ou de répertoires
- Un attachement d'une partition consiste à associer une partition logique d'un disque à un noeud de l'arborescence des fichiers d'une machine

13

Descripteur - File Metadata

- Descripteur de fichier décrit la structure de fichier

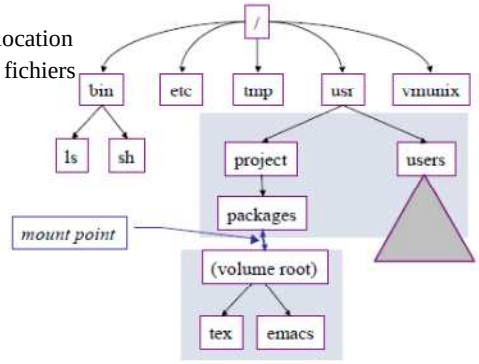


- Chaque opération de fichier (l'ouverture par exemple) doit enregistrer la description du fichier
- Chaque modification de propriétés doit enregistrer dans le disque dur

14

Arborescence des fichiers sous Unix

- Système de fichiers sous Unix est en arbre
- Arbre est représenté par un graph qui contient les répertoires et les fichiers
- Point de montage (*mount point*) est une location physique dans la partition de système de fichiers



Source: J.F. Chase, Université de Duke

Rappel – Commandes de l'Unix

- Localisation dans la hiérarchie : *pwd (le répertoire)*
- Navigation : *cd <destination>*
- Création/destruction de répertoire : *mkdir/rmdir*
- Création de lien symbolique : *ln –s <nom d'origine> <destination>*
- Gestion des droits : *chmod*
- consultation de i-nœuds : *ls –i*

Un exemple :

213345 -rw-r-- r-- 1 wei groupe1 1024 fev 20 15:00 nomfichier1

Inode contient un pointeur vers l'information stockée

Compteur de lien

16

Table d'i-noeuds sous Unix

i-noeud	type
213344	d
213345	f
….	….

→ (nomfichier1, 213345), (nomfichier2, XXXXX),….

→ contenue du fichier nomfichier1

- l'implantation physique des fichiers consiste à attribuer à chaque fichier une structure – i-noeud

17

Montage sous Unix

- Les fichiers différents peuvent être assemblés en un seul : disque physique, disquettes, CD-rom, disque amovible, Zip drive
- Chacun des ces dispositifs est un système de fichiers complet avec une table de i-noeuds par dispositif.
- L'assemblage est fait par l'administrateur avec la commande *mount* qui attache la racine d'un système de fichiers en *un point (mount point)* de l'arborescence d'un autre système (location physique, un disque par exemple).
- Les liens ordinaires ne permettent pas de passer d'un système de fichiers à un autre monté. *Les liens symboliques* le permettent de façon transparente pour l'utilisateur.
- le répertoire */media* est utilisé par le point de montage (*mount point)*

18

Montage sous Unix

- Les fichiers différents peuvent être assemblés en un seul : disques physiques, disquettes, CD-rom, disques amovibles (zip).
- Chacun est un système de fichiers complet, avec une table de i-noeuds par disque.
- L'assemblage est fait par l'administrateur avec la commande *mount*, qui attache la racine d'un système de fichiers en *un point (mount point)* de l'arborescence d'un autre système (location physique, un disque par exemple).
- Les liens ordinaires ne permettent pas de passer d'un système de fichiers à un autre monté. *Les liens symboliques* le permettent de façon transparente pour l'utilisateur
- le répertoire */media* est utilisé par le point de montage (*mount point)*

19

FAT sous Windows

- Rappelons que Windows fait la partition du support physique (disques) par lettres
- Implantation physique des fichiers se fait par FAT (*File Allocation Table*)
- avec FAT, il suffit de connaître le premier bloc correspondant au début du fichier, ensuite de parcourir le tableau jusqu'au bloc voulu par le chaînage
- le nombre « -1 » indique la fin du fichier

Blocs physiques	chaînage
….	….
Fichier1 -> 3	7
….	….
7	8
8	-1
9	….

- exemple: un fichier 1 est localisé par les blocs Physiques 3-7-8 (le chaînage)

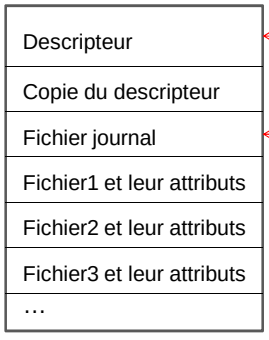
20

NTFS sous Windows (1)

- NTFS (*New Technology File System*) utilisé par Windows NT, Windows 2000 et XP, est un système basé sur une structure appelée «table de fichiers maître », ou MFT *(Master File Table*), permettant de contenir des informations détaillées sur les fichiers
- NTFS est capable de différencier des noms en majuscules de noms en minuscules
- NTFS offre un accès plus rapide que FAT car il utilise une technique plus performants pour localiser les fichiers (un arbre binaire)
- NTFS permet de définir des attributs de **sécurité** pour chaque fichier

21

MFT *(Table des fichiers Maîtres) avec* NTFS sous Windows (2)

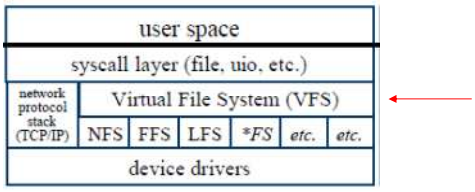


- les informations relatives à chaque fichier sont stockées dans le fichier au sein de la MFT.
- La MFT représente donc une structure de stockage des données de la partition, et non une liste de clusters.

22

VFS : Virtual File System

- Sun Microsystems introduit l'interface VFS en 1985 dans le but de masquer l'accès à un sous-arbre local ou distant de l'arborescence des fichiers ou partition
  - elle est construit sur le concept de V-node
  - elle gère le montage et le démontage de partitions
  - elle permet l'accès aux fichiers lors de traversées de points d'attachement ou points de montage ("mount point")



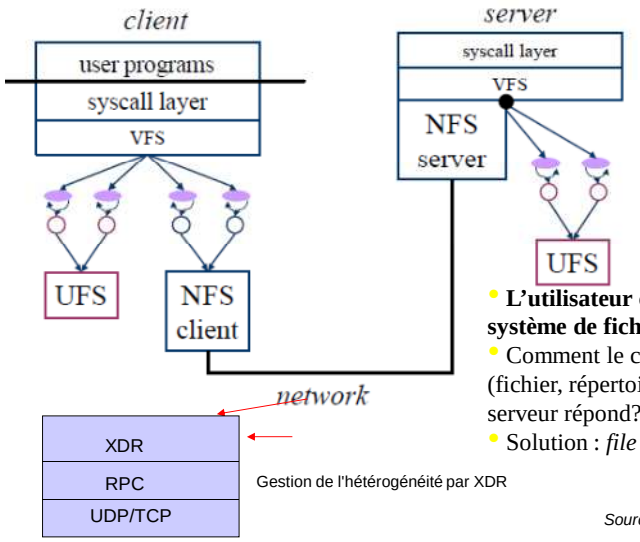
23

Source: J.F. Chase, Université de Duke

Plan

24

Concept général



- **L'utilisateur client peut accès au système de fichiers du serveur**
- Comment le client forme sa requête (fichier, répertoire…)? Comment le serveur répond?
- Solution : *file handle*

Gestion de l'hétérogénéité par XDR

Source: J.F. Chase, Université de Duke

Poignée - File Handle

- **Identificateur** de fichier sur le serveur
- pour le client : un identificateur de 32 octets qu'il n'interprète pas
- pour le serveur : 32 octets dans lesquels il met les informations nécessaires à retrouver un fichier qu'il gère, ou le descripteur de celui-ci

- format d'un identificateur (file handle) avec une taille variable

- par exemple:
<n° de filesystem (8o), n° d'i-node (4o), n° de génération (4o), remplissage (18o)>
- le numéro de génération présente le n° i-nœud d'époque. Ce numéro permet au serveur détecte une anomalie (le client demande un fichier qui peut être détruit par le serveur)
- le file handle se trouve dans *nfs.h*

26

Poignée - File Handle

- Identificateur de fichier sur le serveur
- pour le client : un identificateur de 32 octets qu'il n'interprète pas
- pour le serveur : 32 octets dans lesquels il met les informations nécessaires à retrouver un fichier qu'il gère, ou le descripteur de celui-ci

- format d'un identificateur (*file handle*) avec taille variable

fs id	n° i-node	n° génération i-node	

- par exemple:

<n° de filesystem (8o), n° d'i-node (4o), n° de génération (4o), remplissage (18o)>

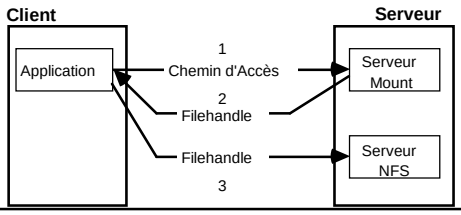
- le numéro de génération présente le n° i-nœud d'époque. Ce numéro permet au serveur détecte une anomalie (le client demande un fichier qui peut être détruit par le serveur)
- le file handle se trouve dans *nfs.h*

27

Protocole *mount*

- Protocole *mount* résout la correspondance entre un nom de répertoire : chemin d'accès sous forme de chaine de caractère qui représente un point d'attachement, et sa localisation sur le serveur cible.

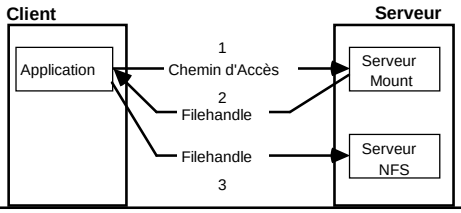
#1 le client envoie un nom de répertoire à attacher au serveur *mount*
#2 le serveur *mount* lui retourne la ***"poignée"*** (file handle) pour pouvoir entrer dans le répertoire d'attachement après avoir vérifié les droits d'accès. C'est une première poignée, et le client la conserve soigneusement sinon, il ne pourra plus accéder au sous-arbre correspondant
#3 Le client, lors d'une opération d'accès à un fichier, envoie au le serveur NFS le file handle qui lui a été retourné pour parcourir le chemin d'accès au fichier qui passe par le point d'attachement



Protocole *mount*

- Protocole *mount* résout la correspondance entre un nom de répertoire : chemin d'accès sous forme de chaine de caractère qui représente un point d'attachement, et sa localisation sur le serveur cible.

#1 le client envoie un nom de répertoire à attacher au serveur *mount*
#2 le serveur *mount* lui retourne la ***"poignée"*** (file handle) pour pouvoir entrer dans le répertoire d'attachement après avoir vérifié les droits d'accès. C'est une première poignée, et le client la conserve soigneusement sinon, il ne pourra plus accéder au sous-arbre correspondant
#3 Le client, lors d'une opération d'accès à un fichier, envoie au serveur NFS le file handle qui lui a été retourné pour parcourir le chemin d'accès au fichier qui passe par le point d'attachement



Protocole *mount* - RPC

- Protocole *mount* est réalisé par un ensemble de procédures RPC SUN du côté serveur. Les principales primitives sont :

NULL	ne fait rien
MOUNT	retourne le file handle (#2 du protocole *mount*)
READMOUNT	retourne la liste des sous-arbres attachés
UNMOUNT	enlève un fichier de la liste des sous arbres attachés
UNMOUNTALL	vide la liste des sous-arbres attachés
READEXPORT	retourne la liste des sous-arbres exportés

30

Protocole NFS

- Protocole *NFS* est *sans état* (stateless). C'est-à-dire, le serveur ne maintient aucune information sur les fichiers qu'il gère pour le compte d'un client. C'est le client qui conserve toutes les informations qui permettent au serveur de retrouver le fichier. Les informations sont dans le *file handle*
- Les appels de RPC sont *synchrones* à cause de « sans état ». Le client lance une requête ; il attend la réponse du serveur (situation bloquante).
- Résistance aux pannes : Quand un serveur tombe en panne, les clients restent bloqués jusqu'à la remise en route du serveur
- Idempotence du service : la sémantique du RPC choisie par Sun implique l'idempotence de toutes les opérations d'accès aux fichiers, par exemple l'écriture séquentielle doit être transformée en écriture en accès direct.

31

Protocole NFS

- Protocole *NFS* est *sans état* (stateless). C'est-à-dire, le serveur ne maintient aucune information sur les fichiers qu'il gère pour le compte d'un client. C'est le client qui conserve toutes les informations qui permettent au serveur de retrouver le fichier, elles sont dans le filehandle
- Les appels de RPC sont *synchrones* à cause de « sans état ». Le client lance une requête; il attend la réponse du serveur (situation bloquante).
- Résistance aux pannes : Quand un serveur tombe en panne, les clients restent bloqués jusqu'à la remise en route du serveur
- Idempotence du service : la sémantique du RPC choisie par SUN implique l'idempotence de toutes les opérations d'accès aux fichiers, par exemple l'écriture séquentielle doit être transformée en écriture en accès direct.

Rappel: une idempotence signifie qu'une opération a le même effet qu'on l'applique une ou plusieurs fois, ou encore qu'en la réappliquant on ne modifiera pas le résultat

32

Opérations (1)

- 1 **GETATTR** (fhandle,attr) rend les attributs d'un fichier
- 2 **SETATTR**(fhandle,attr) définit les attriburs d'un fichier
- 4 **LOOKUP**(dir,name,file,attr) rend un fhandle et les attributs *attr* du fichier *name* dans le répertoire de filehandle *dir*
- 6 **READFILE**(file,offset,count,attr,data) rend le nb des octets de données du fichier « filehandle » dans les données opaques *data*, à partir de la position *offset* dans ce fichier, les attributs attr du fichier sont retournés
- 8 **WRITE**(file,offset,data) écrit dans le fichier de filehandle à partir de la position offset, les données opaques data transmises, l'opération write est atomique
- 9 **CREATE**(dir,name,satt,file,attr) crée un fichier de nom *name* dans le répertoire de file handle *dir* avec les attributs *satt*, le file handle *file*, et les attributs du fichier créé sont retournés
- ….

33

Opérations (2)

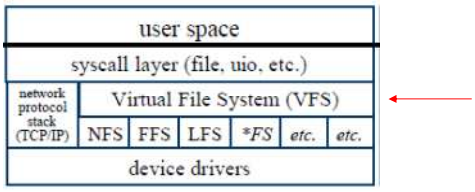
- NFS3 définit principalement les nouvelles procédures (l'accès sécurisé, la recherche du répertoire, la sécurité asynchrone)
  - 4 **ACCESS**(object, access) permet de vérifier le droit d'accès à l'objet
  - 3 **LOOKUP** (dir, name, object, objattr, dirattri) rend le filehandle et les attributs attr du fichier name(object) dans le répertoire de filehandle dir et les attributs attr
  - 21 **COMMIT** (file, offset, count, filewcc, verf) permet de valider l'écriture du fichier de la cache à la mémoire stable (soft asynchronous)

- Parmi les paramètres,
  - *verf* est un cookie qui permet au client de déterminer si le serveur a fait un reboot entre la procédure WRITE et celle COMMIT
  - *filewcc* concerne les attributs du fichier

34

VFS : Virtual File System

- Sun Microsystems introduit l'interface VFS en 1985 dans le but de masquer l'accès à un sous-arbre local ou distant de l'arborescence des fichiers ou partition
  - elle est construit sur le concept de V-node
  - elle gère le montage et le démontage de partitions
  - elle permet l'accès aux fichiers lors de traversées de points d'attachement ou points de montage ("mount point")

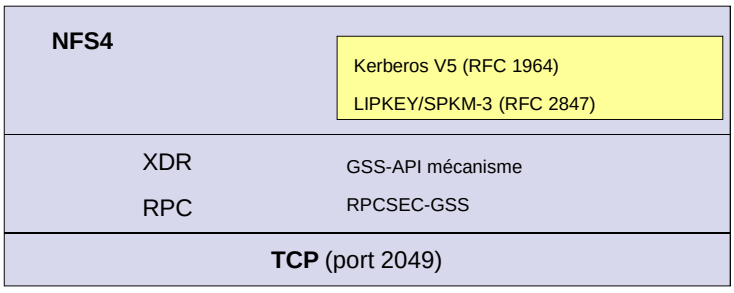


35

Source: J.F. Chase, Université de Duke

Concept - NFS version 4

- **Quoi de nouveaux?**
- La sécurité renforcée assure que chaque requête est en sécurité (authentification et intégrité)
- Requêtes composées à la place de plusieurs requêtes (*compound operation*)
- Les migration et réplication
- une seule espace de nom de fichiers….



36

NFS4 – sécurité renforcée

- Le Kerberos est un protocole d'authentification du type client-serveur
  - Clé privée est traitée par les techniques LIPKEY/SPKM-3
  - Clé public est traitée par la technique Kerberos V5
- Les interfaces de services sécurisés
  - GSSAPI (Generic Security)
  - RPCSEC_GSS
- La négociation des paramètres entre le client et le serveur
- La première poigné (*filehandle*) est public ou root

37

NFS4 – opérations composées

- La procédure (*compound procedure*) permet de regrouper plusieurs opérations (requêtes) dans une seule opération afin de réduire les appels RPC. Onestime qu'une requête composée de la version 4 du protocole exige cinq fois moins d'interactions client-serveur qu'avec la version 3.

GETATTR
ACCESS
LOOKUP
READ

quatre requêtes avec version **3**

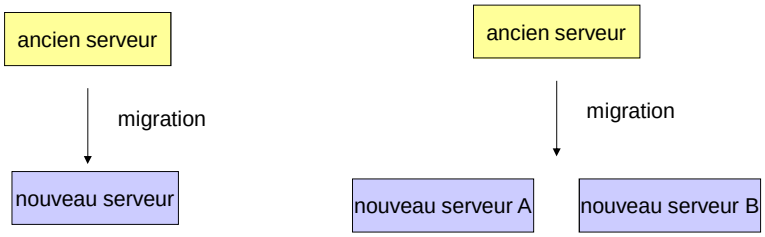
GETATTR
ACCESS
LOOKUP
READ

une requête avec version **4**

38

NFS4 – migration (1)

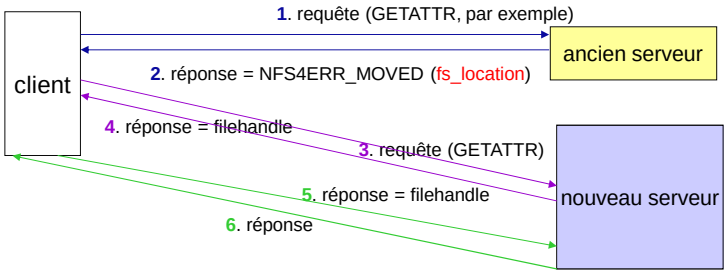
- La migration d'un système de fichiers à un autre repose sur le remplacement (*location* en anglais) ou plusieurs remplacements. Un remplacement « *fs_location*» contient deux paramètres : le nom du nouveau serveur et la racine de répertoire (*root*)



39

NFS4 – migration (2)

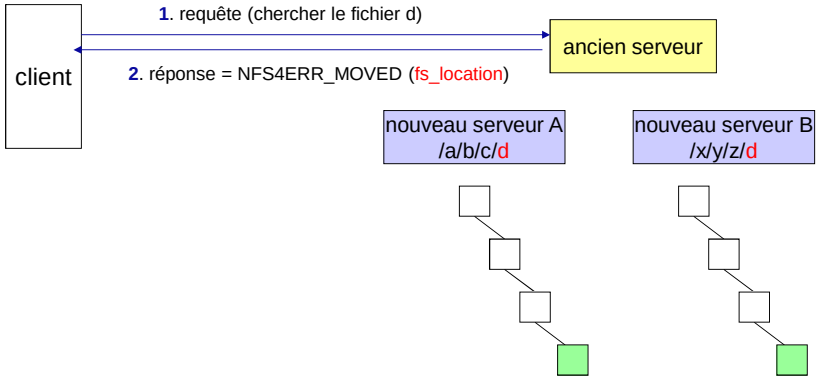
- La requête sur l'ancien serveur est dirigée vers le nouveau serveur par les six étapes :



40

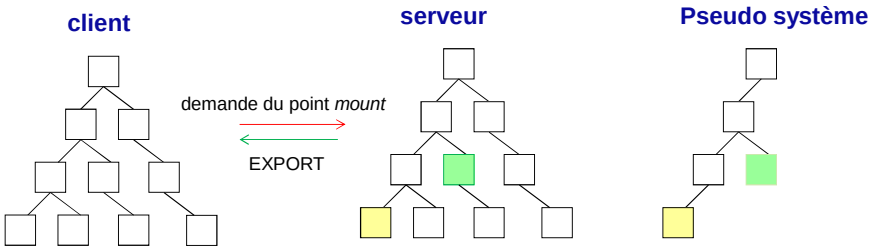
NFS4 – réplication

- Si le fichier *d* est répliqué dans le serveur A et le serveur B, *fs_location* doit indiquer la réplication (serveurs A et B) et les accès aux fichiers (/a/b/c/d et /x/y/z/d)



NFS4 – Pseudo système de fichiers

- «Exports» de la procédure *mount* permet de créer une image de l'arborescence des fichiers du serveur. Donc, une seule espace de nom de fichiers. Les fichiers ne sont pas « exportables » restent comme un pseudo système.



42

NFS4 – cache du côté client

- Comment assurer une cohérence de données sachant que chaque client n'a qu'une connaissance de son propre cache ?
- Avec NFS2 et 3, le client peut demander au serveur de bloquer un fichier (LOCK) afin d'écrire ou de lire ce fichier
- NFS4 ne définit pas une cohérence de caches distribués, mais
- il limite d'utilisation des opérations LOCK et SHARE
- Il introduit le mécanisme de « délégation ». Le serveur délègue le client pour faire certaines décisions. Par exemple, le client peut **lire** un fichier sans l'accord du Serveur jusqu'à un autre client veut également lire le fichier en question. -> le mécanisme « délégation » évite que le serveur reçoit les mêmes requêtes répétées.
- De toute façon, le client devrait contacter le serveur en vue de la cohérence de données

43

Plan

44

Configuration

Du côté serveur : modifier le fichier /etc/rc.conf par
rpcbind_enable = « YES » //bind permet le client de trouve le serveur
nfs_server_enable = « YES »
mountd_flags=« -r »

Du côté client : modifier le fichier /etc/rc.conf par
nfs_client_enable = « YES »

Le fichier /etc/exports doit indiquer comment exporter des fichiers. Par exemple :
export *src* and *ports* to client01 and client02, but only # client01 has root privileges on it /usr/src /usr/ports -maproot=root client01 /usr/src /usr/ports client02

45

Conclusion

- NFS est un protocole de systèmes de fichiers initialisé par Sun Microsystem
- XDR effectue la gestion de données (hétérogénéité)
- RPC réalise les appels à distance client-serveur

- Les différentes versions de NFS évoluent vers un protocole accessible à via TCP/IP ou UDP/IP
- Les différentes versions de NFS évoluent également par les nouvelles opérations (COMPOUND de NFS4, par exemple)
- Les différentes versions de NFS évoluent dans le but de passer à l'échelle (nb de clients augmenté et l'extension de distance).
- La technologie « mirroring » (mêmes systèmes chez un serveur) n'est pas définit par NFS4

- la sécurité est renforcée par la version 4. Donc, elle rend la migration et la réplication plus efficaces

46