

7 juin 2006

Nicolas Toper  
Ingrid Roussel

Systemes de gestion de fichiers  
de grandes tailles largement  
distribués – 2eme partie

intro

---

# Plan

---

- I. Concepts généraux
- II. Passage à l'échelle
- III. Sécurité
- IV. Etude de cas: S3

## **NB**

- Exposé avec commentaire disponible en PDF sur <http://www.deviant-abstraction.net/2006/06/02/large-scale-distributed-file-system-2/>

# Concepts généraux

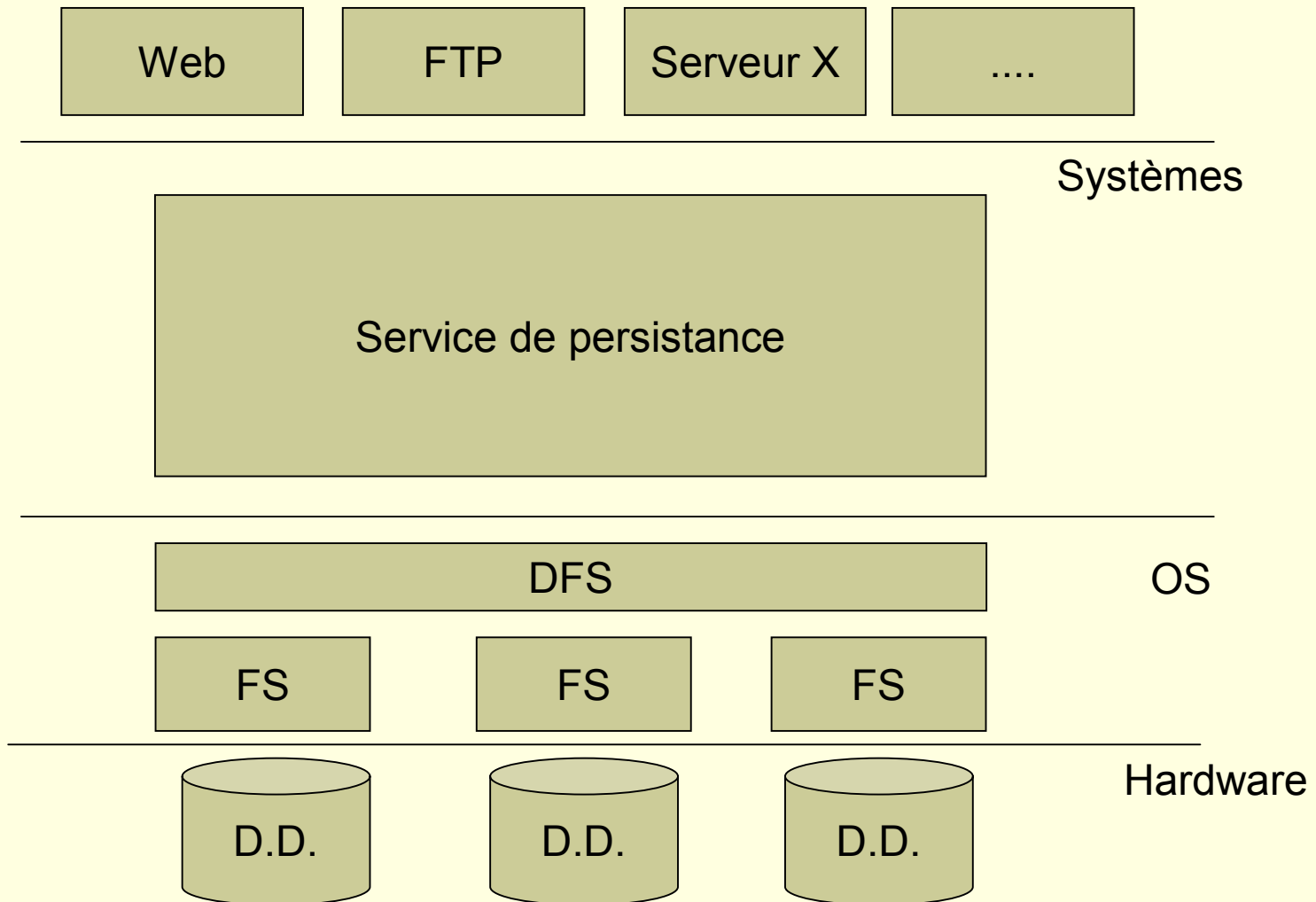
---

# Concepts généraux: définitions d'un DFS

---

- FS (système de fichier)
  - Méthode d'organisation des données persistantes sur un medium de stockage de masse et durable.
  - Abstraction du service de persistance
- Distribuer le FS (DFS – système de fichiers distribué)
  - Objectif initiale: partage de fichiers
  - Actuellement:
    - Gestion de très grands systèmes de fichiers (très grand volume de données)
    - Augmenter fonctionnalités par parallélisation (tolérance aux pannes, performance, passage à l'échelle)
    - Factoriser fonctionnalités de distribution (évite aux couches supérieures de les implanter)

# Concepts généraux: architecture système d'un DFS



# Concepts généraux: Classes de fichiers

---

- *Accès en lecture (implicite)*
- Accès en écriture
- *Accès séquentiel (implicite)*
- Accès aléatoire
- *Accès concurrent lâche (implicite)*
- Accès concurrent critique

# Concepts généraux: fonctionnalités

---

- Espace disque illimité utilisateur
- Accès aux données de n'importe où
- Interface d'accès aux données évoluée
- Confidentialité des données
- Versioning et archivage des données
- Tolérants aux fautes
- Mode déconnecté
- Coût

# Concepts généraux: problèmes

---

- Latence importante, débit faible et ce sans gigue
- Sécurité
- Passage à l'échelle d'Internet
- Performance
- Tolérance aux pannes et sûreté de fonctionnement
- Gestion des caches et répliques
- Gestion des différentes classes de fichiers

# Passage à l'échelle

---

# Passage à l'échelle: définitions

---

- **Aspect taille du système** ajouter facilement des utilisateurs et/ou des ressources au système.
- **Aspect géographique** les utilisateurs et les ressources sont dispersés géographiquement
- **Aspect administratif** le système est sur plusieurs domaines administratifs différents
- **Aspect économique**  $S = R/C$
- **Autres aspects.**

# Passage à l'échelle et performance

---

- Nécessité d'un SLA
- Instabilité des ressources du système
- Goulets d'étranglement
  - Latence réseau
  - Service centralisé
- Solutions
  - Communication asynchrone
  - Distribution et répliques
  - Cache.

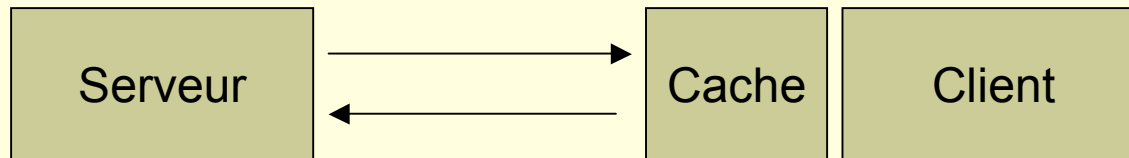
# Exemple de passage à l'échelle: Pond et NFS

<i>Phase</i>	<i>LAN</i>		<i>WAN</i>		<i>Predominant operations in benchmark</i>
	<i>Linux NFS</i>	<i>Pond</i>	<i>Linux NFS</i>	<i>Pond</i>	
1	0.0	1.9	0.9	2.8	Read and write
2	0.3	11.0	9.4	16.8	Read and write
3	1.1	1.8	8.3	1.8	Read
4	0.5	1.5	6.9	1.5	Read
5	2.6	21.0	21.5	32.0	Read and write
Total	4.5	37.2	47.0	54.9	

(en ms)

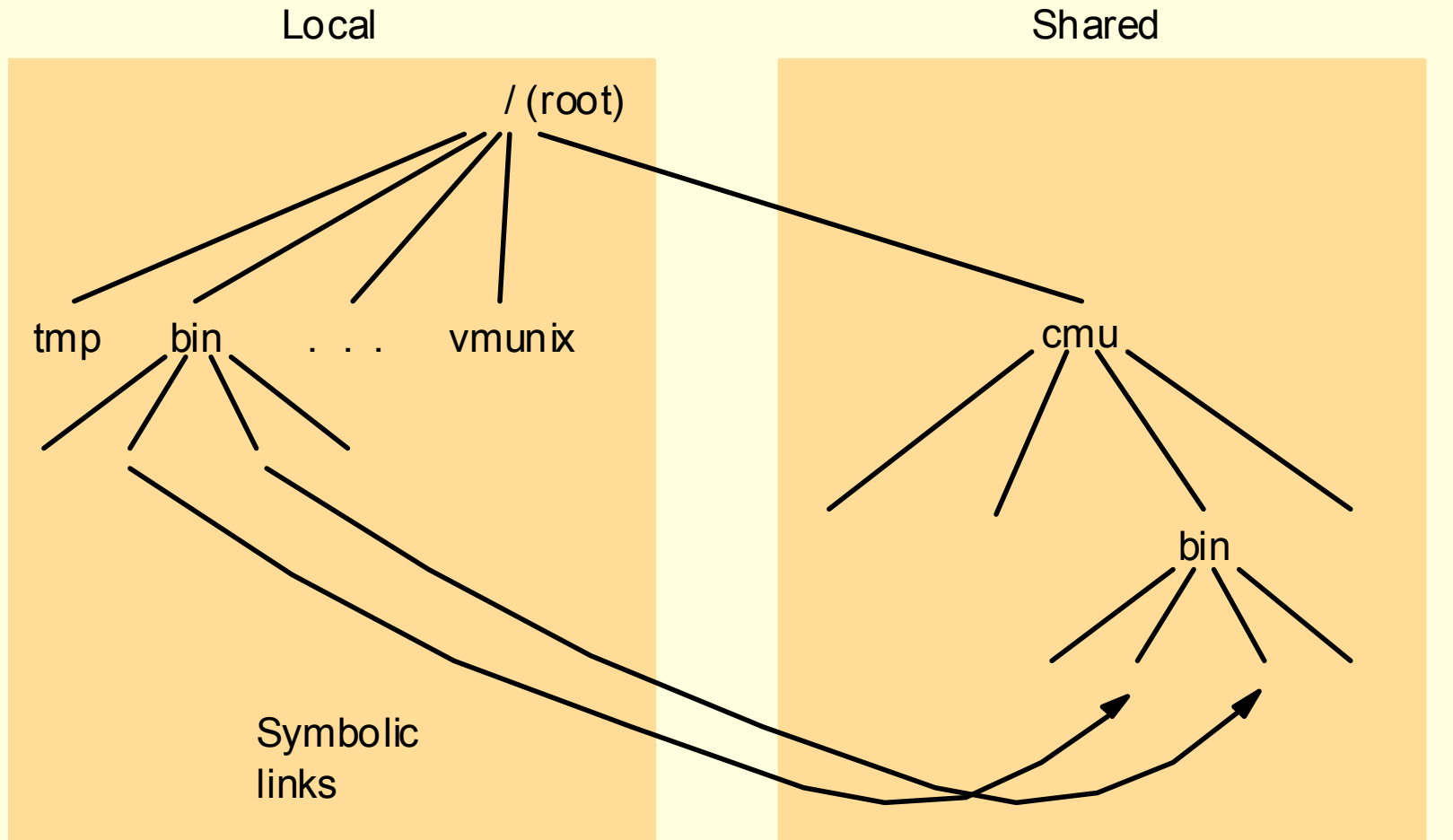
(source: Coroulis)

# Passage à l'échelle et cache



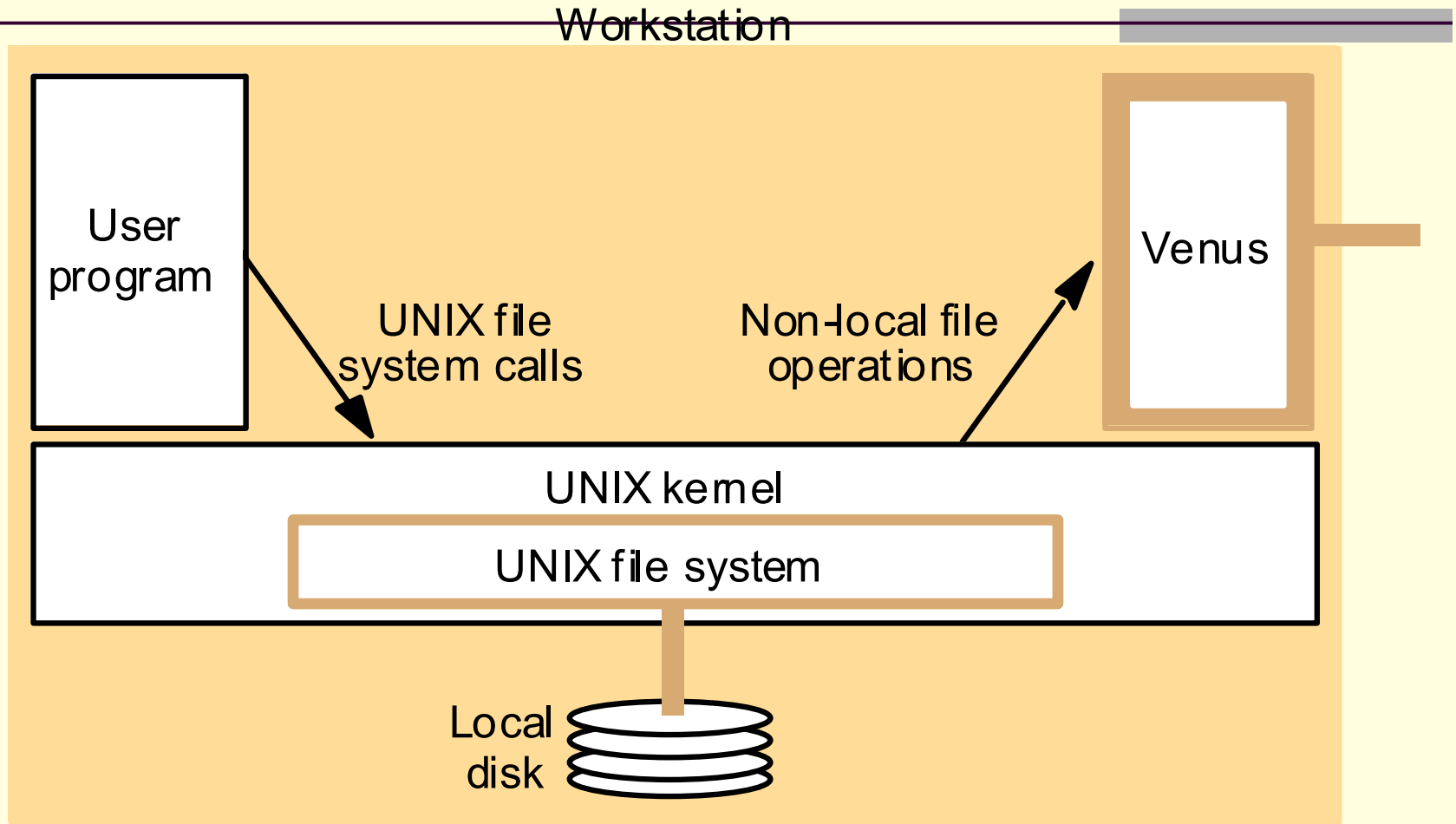
- Exploite principe de localité des références
- $(T - T_c < t) \vee (T_{client} = T_{serveur})$
- Cohérence des caches
  - Détection: avant, pendant ou après E/S
  - Gestion: pull/push; write-back/write-through
- Suppression des entrées: LRU ou FIFO
- Efficacité d'un cache?

# Passage à l'échelle: Vue utilisateur d'AFS



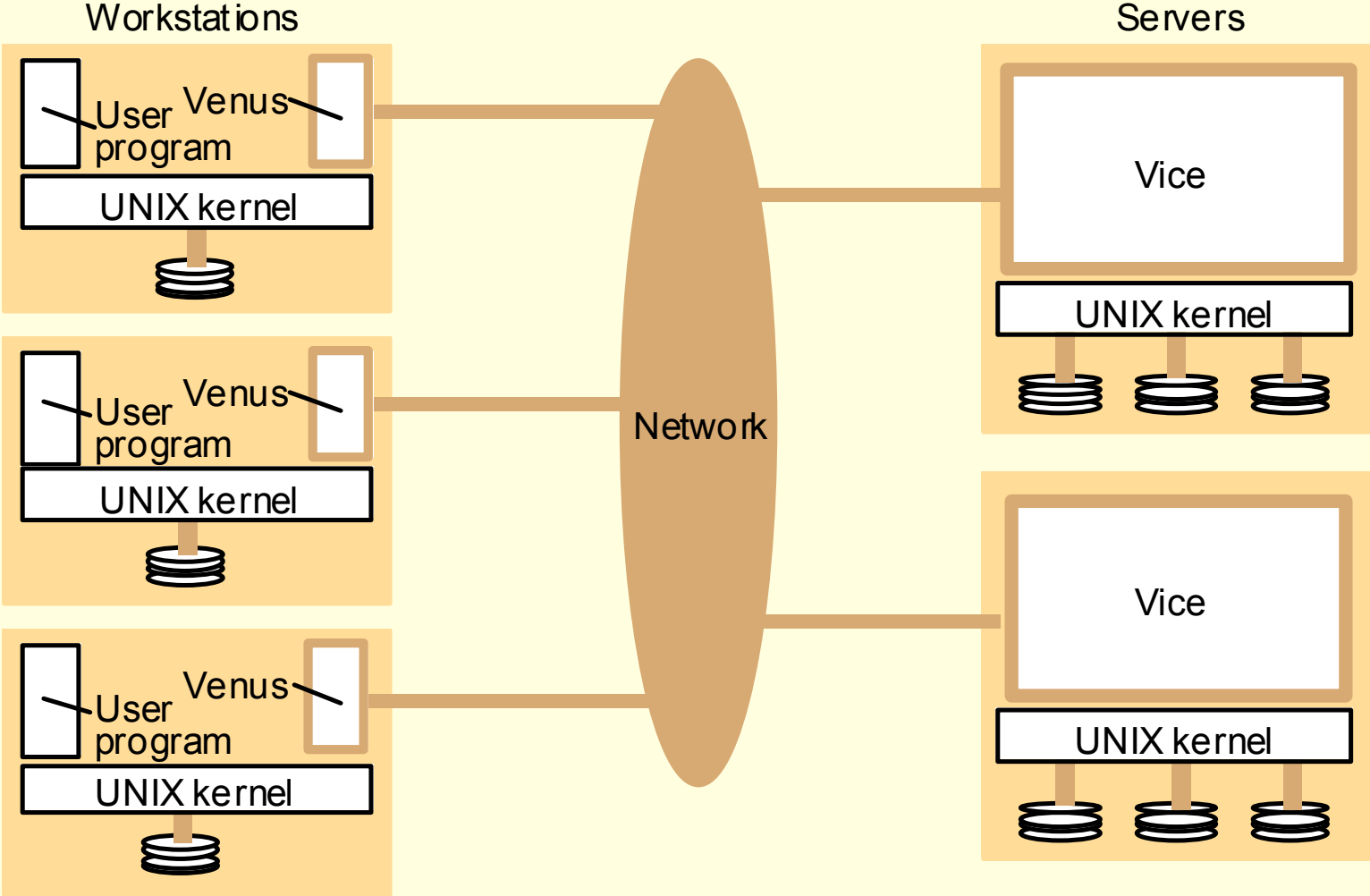
(source: Coroulis)

# Passage à l'échelle: Vue système d'AFS

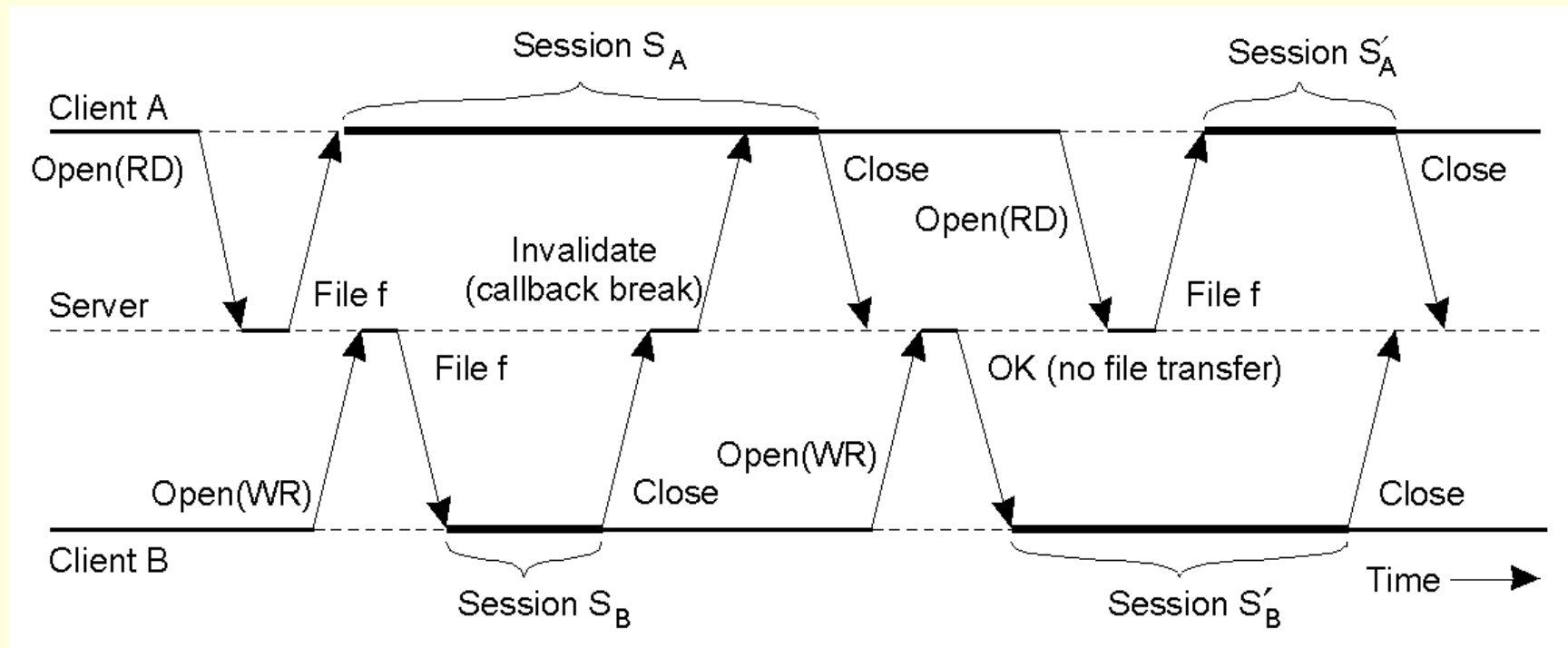


(source: Coroulis)

# Passage à l'échelle: Architecture d'AFS



# Passage à l'échelle: Exemple de fonctionnement du cache AFS



(source: Tanenbaum)

# Passage à l'échelle: Formalisation du cache d'AFS

---

- **Open réussi** -> latest(F,S, 0) v  
(lostCallback(S,T) && inCache(F) && latest(F,S,T))
- **Open échoué** ->failure(S)
- **Close réussi** -> updated(F,S)
- **Close échoué** -> failure(S)
- AFS créé pour offrir un passage à l'échelle et de bonnes performances

# Passage à l'échelle: Nécessité de la tolérance aux pannes

---

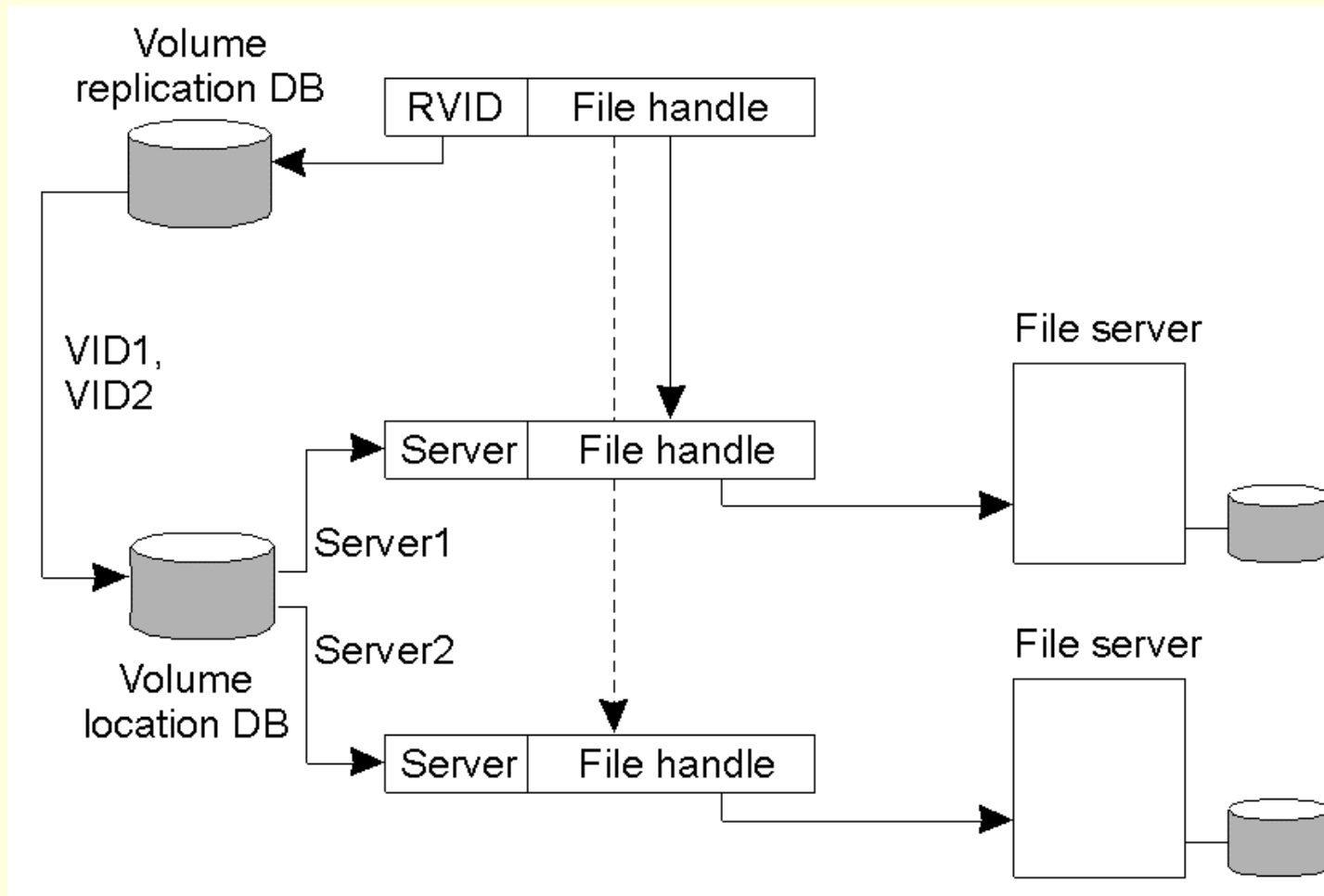
- n éléments actifs du DFS en série.  $P(x)$ :
- Fiabilité: probabilité pour qu'un système soit continûment en fonctionnement sur une période donnée (entre 0 et t). Noté R
- $R(x)$  fiabilité de chaque élément x ( $R(1)$ ,  $R(2), \dots, R(n)$ )
- $R(\text{DFS}) = [i: 1 \text{ à } n] \prod R(i)$
- $[n \rightarrow +\infty] \text{Lim } (R(\text{DFS})) = 0$
- Système en parallèle:  
 $[n \rightarrow +\infty] \text{Lim } (R(\text{DFS})) = +1$

# Passage à l'échelle: Techniques de tolérance aux pannes

---

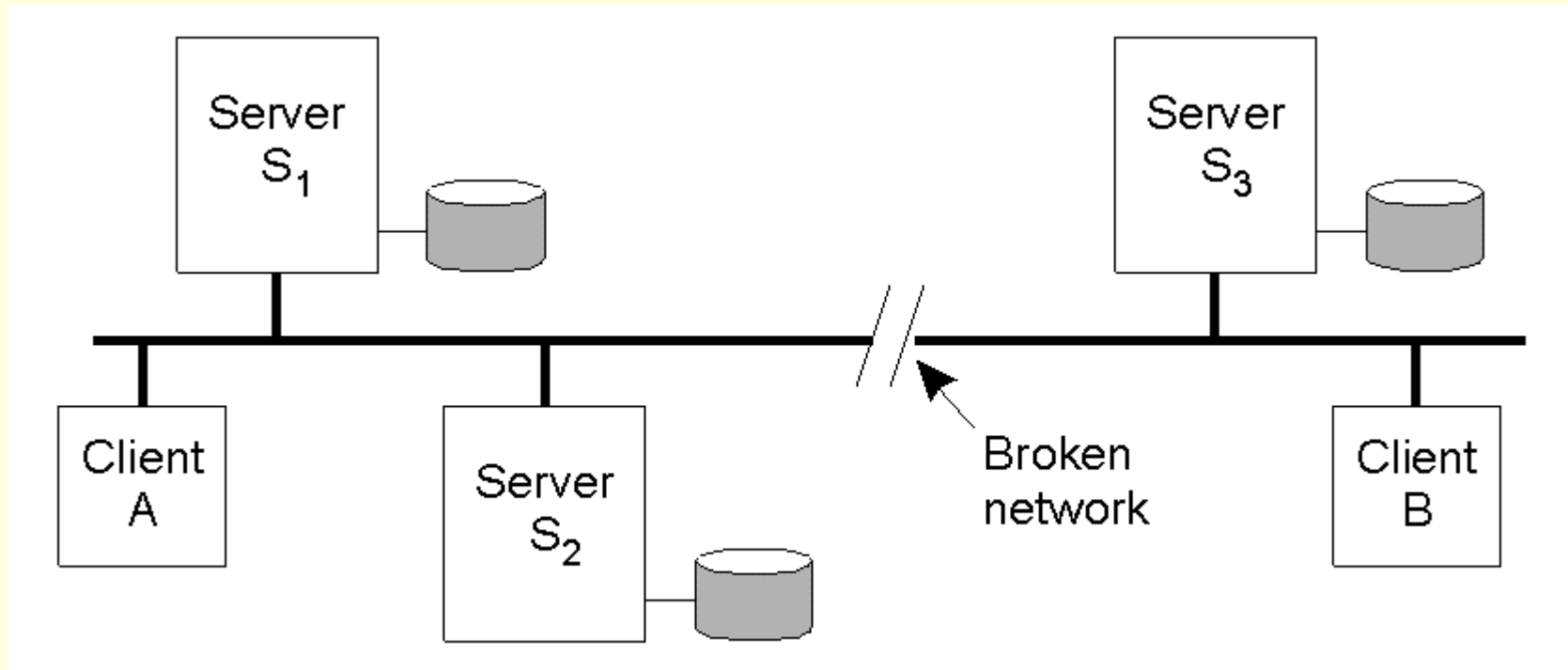
- DFS est un service de persistance
- Typologies des pannes sur un DFSol
  - Franche, transitoire, intermittente, temporelle, byzantines
  - Serveur, client, réseau
  - Lecture, écriture, stockage
- Solutions
  - Réplication
  - Fragmentation
  - Points de reprise

# Passage à l'échelle: Incidence de la réplication sur le nommage dans Coda



(source: Tanenbaum)

# Passage à l'échelle: Partition du réseau en environnement répliqué dans Coda



(source: Tanenbaum)



# Passage à l'échelle en peer-to-peer

---

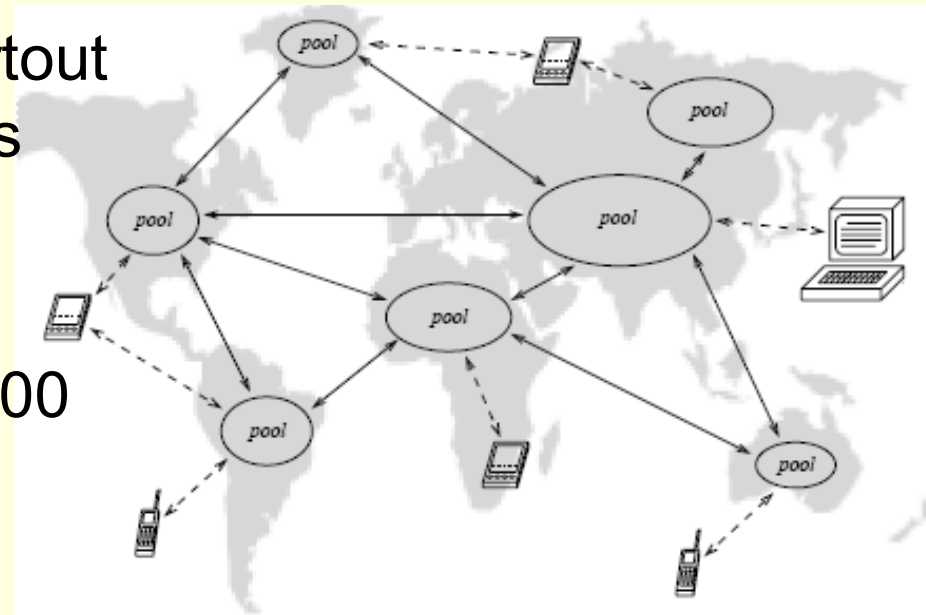
- Présentation d'OceanStore
- Redondance des données
- Localisation et Routage



# OceanStore

## Systeme de Stockage distribue

- Espace de stockage P2P
- Données consultables partout et par plusieurs utilisateurs
- Non accessibles aux utilisateurs non autorisés
- $10^{10}$  utilisateurs avec 10.000 fichiers chacun
- Fragmentation : fichier fragmenté / transféré / dupliqué : Codage Reed-Solomon



(source: Site de Berkeley)



# Oceanstore :

## Localiser une donnée

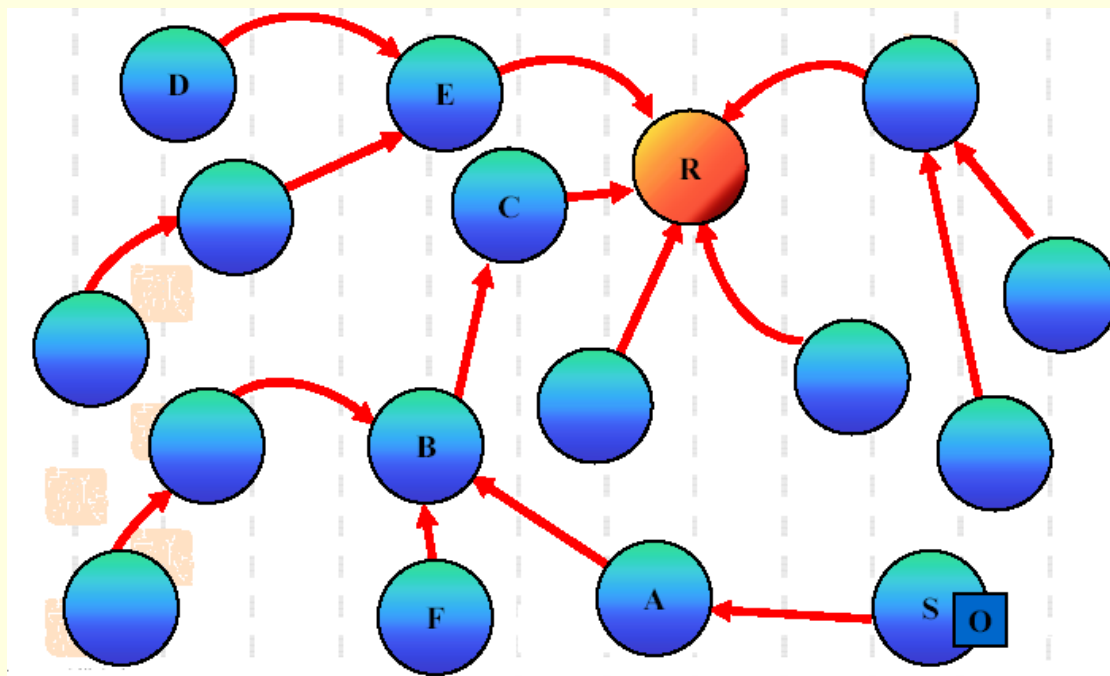
---

- Impossibilité d'avoir une vue globale de l'ensemble des ressources, et difficulté de retrouver une ressource particulière
- **Tapestry :**
  - Nommage : Pairs non plus représentés par une adresse IP mais par un Identifiant défini dans l'Overlay : GUID (hachage SHA-1)
  - Localisation et Routage effectués en même temps



# Oceanstore

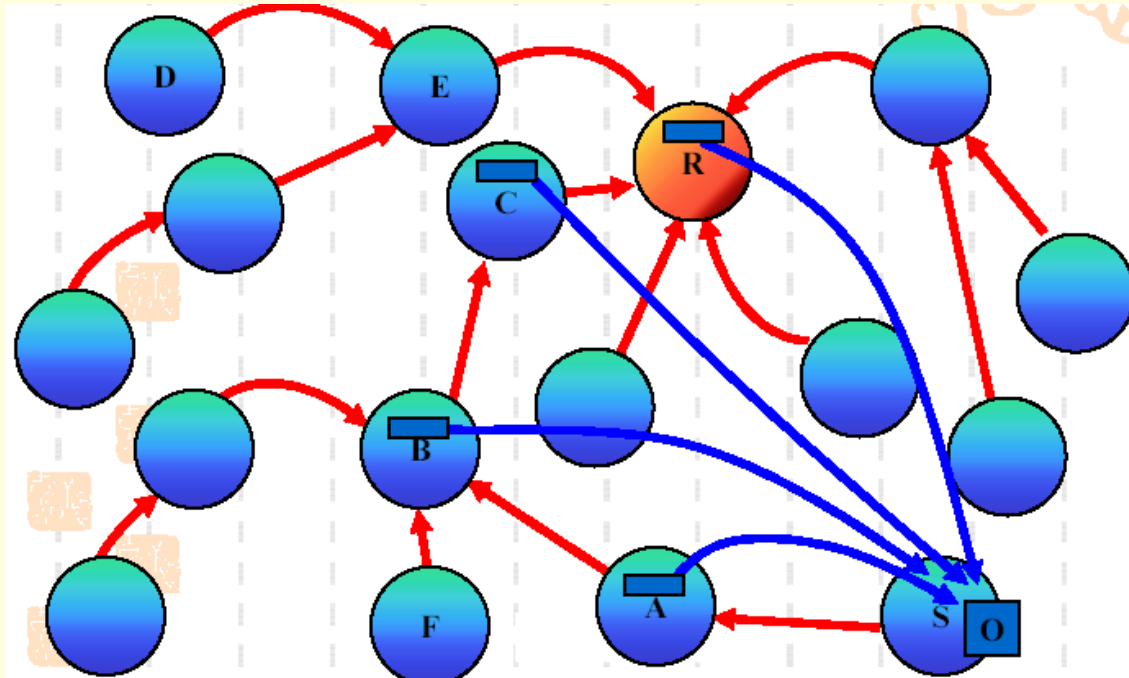
## Publication / Localisation (1/3)



- Chaque ID est rattaché à 1 nœud racine
- Message de publication du Serveur possédant l'objet vers le nœud racine

# Oceanstore

## Publication / Localisation (2/3)

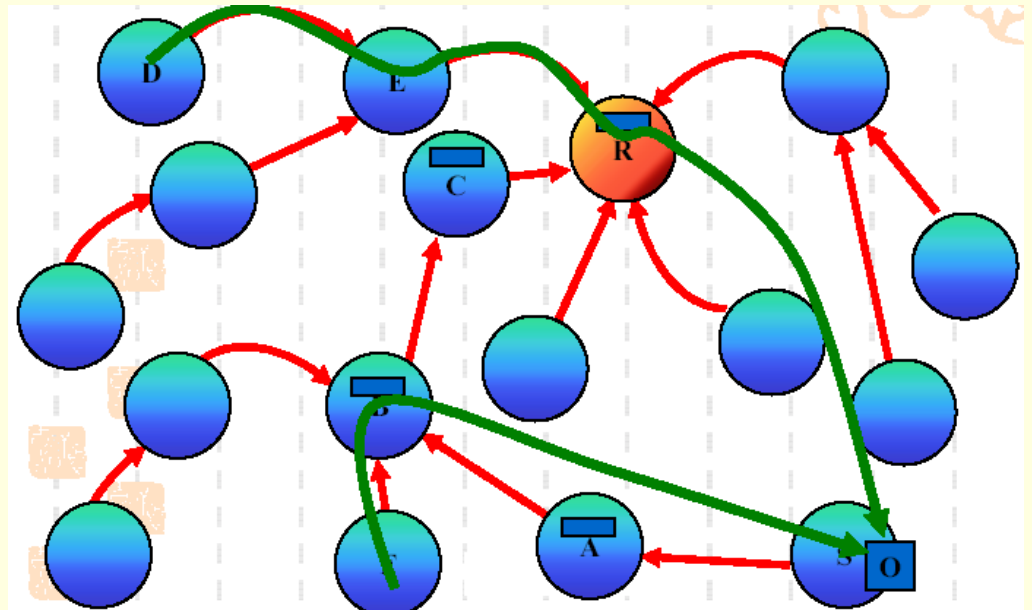


- A chaque saut, les nœuds stockent des pointeurs vers la source
- ObjetID/ServerID

# Oceanstore

## Publication / Localisation (3/3)

Requête de localisation



Un nœud veut localiser un objet O donc fait une requête vers le nœud racine :

- Si un nœud intermédiaire possède le pointeur, alors la requête est redirigée vers le nœud racine
- Sinon le nœud racine se charge de cette opération



# OceanStore :

## Localisation et Routage (Algorithme de Plaxton)

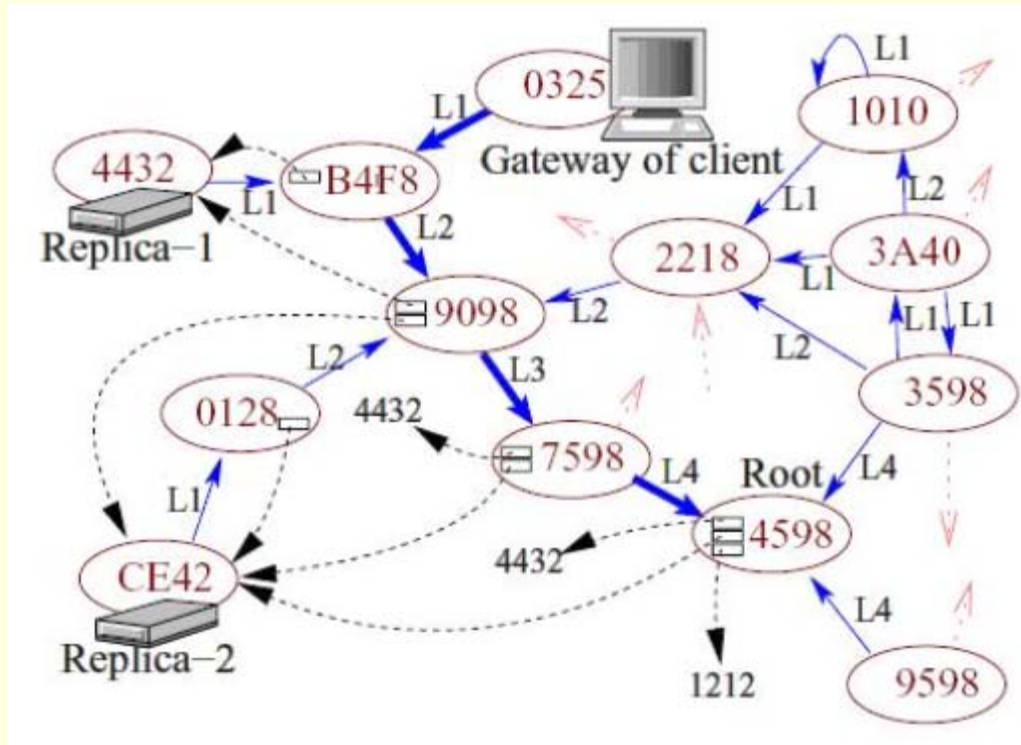


Table de routage :

	0	1	...	E	F
N1	xxx0	xxx1	...	xxxE	xxxF
N2	xx05	xx15	...	xxE5	xxF5
N3	x025	x125	...	xE25	xF25
N4	0325	1325	...	E325	F325

(source: Site de Berkeley)

# Passage à l'échelle: Conclusion

---

- Passage à l'échelle est complexe: plusieurs dimensions à traiter
- Pour gérer les performances: cache et communications asynchrones
- Choix d'une cohérence structurant surtout pour la tolérance aux pannes.
- Passage à l'échelle entraîne des pertes de performance
- Passer l'échelle d'Internet?

# Sécurité

---

# SFS (Self-certifying File System)

## Architecture d'authentification SFS

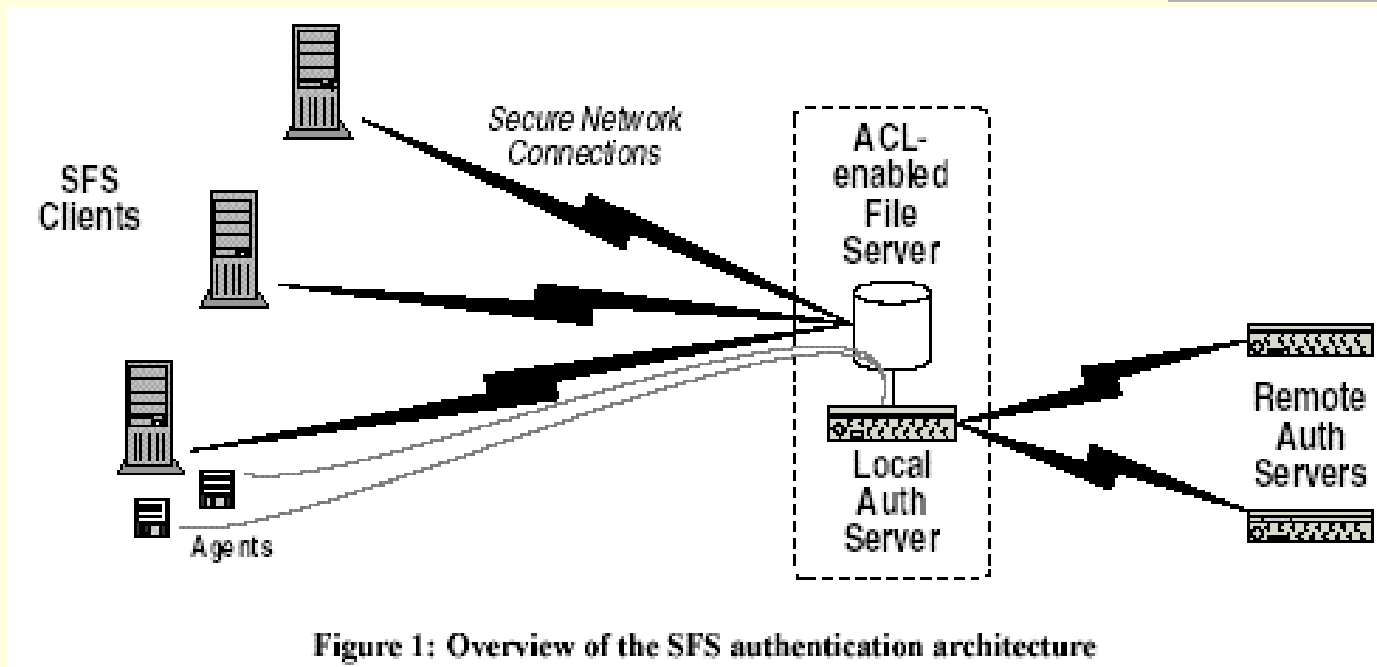


Figure 1: Overview of the SFS authentication architecture

(source : Site de Berkeley)

- Authentification, Autorisation
- Établissement d'une relation de confiance entre domaines



# SFS (Self-certifying File System)

## Nommage des utilisateurs et des groupes

---

- Hachage des clés publiques
- Nom des utilisateurs
- Nom des groupes
  
- Exemple :
  - Noms locaux
    - tony
    - asdof876a4we7p7oif
  - Noms distants
    - tony@db.uwaterloo.ca
    - Asdof876a4we7p7oif@db.uwaterloo.ca



# SFS (Self-certifying File System)

## ACL

- Une ACL spécifie les droits d'accès aux fichiers pour un utilisateur ou un groupe
- Chaque fichier contient une ACL (située dans les 512 premiers bits d'un fichier)
- Exemple d'ajout de membres dans un nouveau groupe

```
$ sfskey group \  
-m +u=james \  
-m +u=liz@bu.edu,gnze6... \  
-m +g=students@mit.edu,w7abn9p... \  
-m +p=anb726muxau6phtk3zu3nq4n463mwn9a \  
charles.cwpeople
```



# SFS (Self-certifying File System) Structure

- Chemins de fichiers auto-certifiés avec une clé de hash  
Ex. : /sfs/hote:host\_id [host\_id = hash (hôte, clé publique du serveur)]



(source: Tanenbaum)

- **LOC** : localisation qui correspond soit au nom de domaine DNS du serveur SFS (ou son @ IP). SFS suppose que tous les serveurs ont une clé publique.
- **HID** (Host Identifier) : obtenu par un hash crypté de la localisation et de la clé publique
- **PATHNAME** : Chemin local où le fichier se situe

# Etude de cas: S3

---

# S3: Présentation

---

- S3: object store distant
- Service de persistance.
- « Base » d'un DFSol
- Paiement à l'usage
- Basé sur les Web services
- Concept de bucket
- Objet = données + méta-données
- Partage d'objets
- Clés.

# S3: Exemple REST

## ■ Requête

```
PUT /quotes/Neo HTTP/1.0
x-amz-meta-family: Anderson
Content-Length: 4
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+F1c8F2Cy8=
Date: Thu, 17 Nov 2005 07:48:33 GMT
Content-Type: text/plain
```

woah

## ■ Réponse

```
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAilfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wilO1xl+zbwZ163pt7
x-amz-meta-family: Anderson
x-amz-request-id: 0A49CE4060975EAC
Date: Thu, 17 Nov 2005 07:48:32 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Length: 0
Connection: close
Server: AmazonS3
```

# S3: Méthodologie de tests

---

- Objectif: évaluation de la performance de S3
- Utilisation de s3Shell
- Utilisation de l'instrumentation fournie
- Calcul du temps nécessaires pour stocker et récupérer un fichier
- REST sur HTTP
- Opérations de tests: stockage, récupération d'un fichier et listing du bucket.

# S3: Résultats (1/2)

---

- Fichier test mp3 de 5 151 Ko
- Transmission par HTTP au bout d'une ligne ADSL
- Stockage du fichier : 2min 52.422s
- Stockage du fichier transmis compressé (Zlib): 2min 52.782s
- Téléchargement : 43.344s
- Téléchargement compressé: 44.125s
- List (2 éléments): 1.375 s
- Autres opérations: environ 1s

# S3: Résultats (2/2)

---

- `time ./s3curl.pl --id=[myid] --key=[mykey] --put /data/install/thunderbird-1.0.6.tar.gz -- http://s3.amazonaws.com/\[mybucket\]/thunderbird-1.0.6.tar.gz`  
**real 0m2.312s**  
**user 0m0.170s**  
**sys 0m0.134s**
- `time ./securl.pl --id=[myid] --key=[mykey] -- http://s3.amazonaws.com/\[mybucket\]/thunderbird-1.0.6.tar.gz -o thunderbird-1.0.6.tar.gz`  
**real 0m3.725s**  
**user 0m0.231s**  
**sys 0m0.266s**

# Conclusion

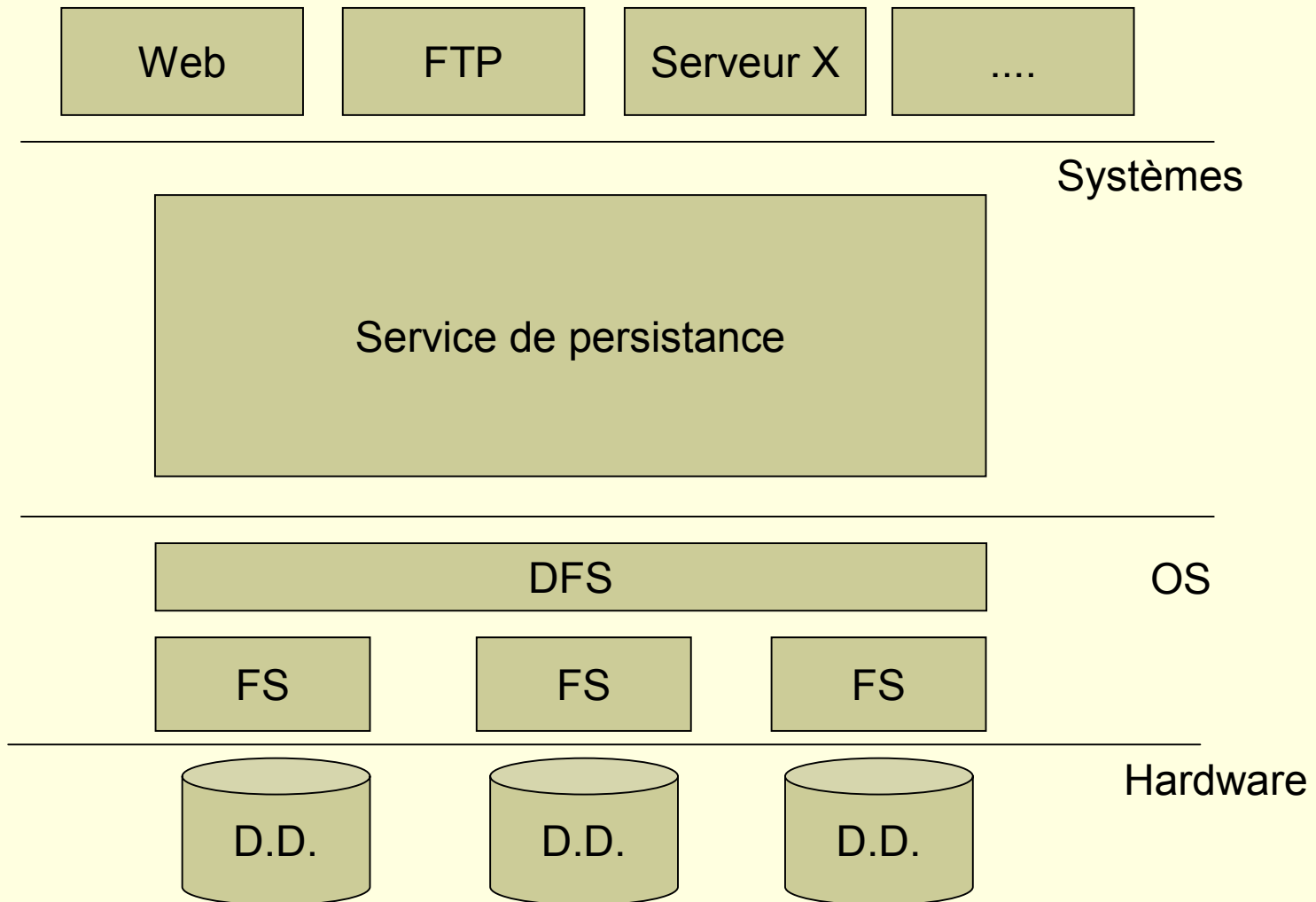
---

# Concepts généraux: modèles de conception

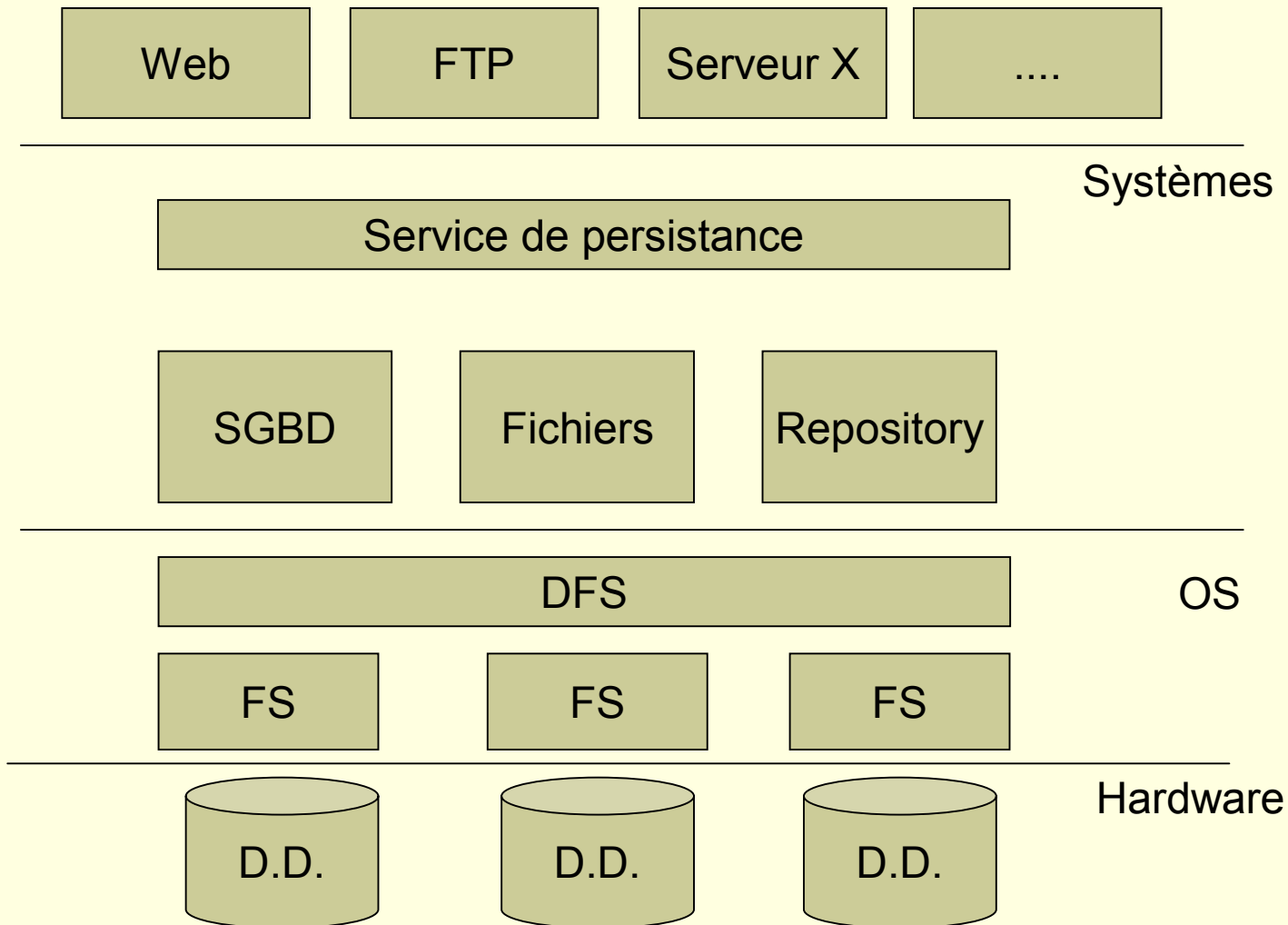
---

- Système largement distribué
- Gestion de classes de fichiers
- Cryptage
- Cache
- Deux approches du stockage de fichier:
  - Un fichier sur une machine
  - Un fichier sur plusieurs machines
- Localisation: (ID +) @ machine + @ locale
- Communication asynchrone

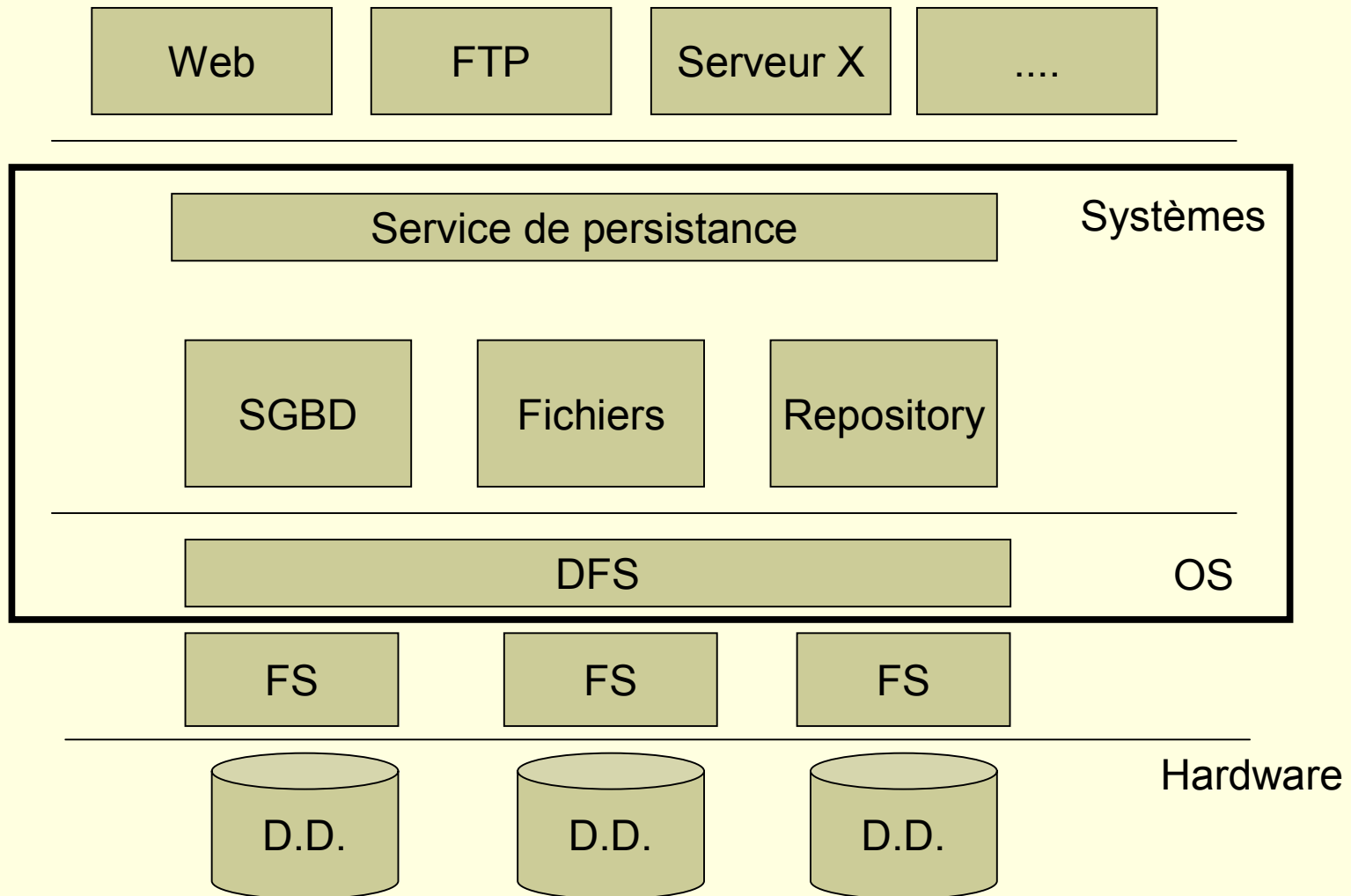
# Conclusion: architecture système d'un DFS (1/3)



# Concepts généraux: Vue détaillée du service de persistance (2/3)



# Concepts généraux: architecture système d'un DFS sur Internet (3/3)



# Annexes

---

# Bibliographie

---

- Coulouris, **Distributed Systems: Concepts and Design**, 2005
- Tanenbaum, **Distributed Systems: Principles and Paradigms**, 2002

# Webographie

---

- <http://aws.amazon.com> pour S3
- <http://www.allthingsdistributed.com/>, Werner Vogels, CTO Amazon
- <http://www.coda.cs.cmu.edu/> pour Coda
- <http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/SAR/> cours d'E. Gressier sur la cohérence
- <http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/SAR/> cours de G. Florin sur la tolérance aux pannes
- <http://www.coda.cs.cmu.edu/doc/ppt/server.ppt> structures de données de Coda.
- <http://www.coda.cs.cmu.edu/doc/html/coda-howto.html#toc5> FAQ de Coda
- [http://en.wikipedia.org/wiki/List\\_of\\_file\\_systems](http://en.wikipedia.org/wiki/List_of_file_systems) Une liste de système de fichiers
- <http://oceanstore.cs.berkeley.edu/> lien officiel sur OceanStore

# Concepts généraux: principaux produits présentés

---

- AFS et Coda
  - Projets universitaires
  - Passage à l'échelle
  - Haute disponibilité
- OceanStore
  - Projet universitaire
  - Stockage P2P
  - Fichiers immutables
- S3 d'Amazon
  - Produit industriel
  - Basé sur HTTP et BitTorrent
  - API Web Services et REST

# Concepts généraux: présentation d'autres produits

Issue	NFS	Coda	Plan 9	xFS	SFS
Design goals	Access transparency	High availability	Uniformity	Serverless system	Scalable security
Access model	Remote	Up/Download	Remote	Log-based	Remote
Communication	RPC	RPC	Special	Active msgs	RPC
Client process	Thin/Fat	Fat	Thin	Fat	Medium
Server groups	No	Yes	No	Yes	No
Mount granularity	Directory	File system	File system	File system	Directory
Name space	Per client	Global	Per process	Global	Global
File ID scope	File server	Global	Server	Global	File system
Sharing sem.	Session	Transactional	UNIX	UNIX	N/S
Cache consist.	write-back	write-back	write-through	write-back	write-back
Replication	Minimal	ROWA	None	Striping	None
Fault tolerance	Reliable comm.	Replication and caching	Reliable comm.	Striping	Reliable comm.
Recovery	Client-based	Reintegration	N/S	Checkpoint & write logs	N/S
Secure channels	Existing mechanisms	Needham-Schroeder	Needham-Schroeder	No pathnames	Self-cert.
Access control	Many operations	Directory operations	UNIX based	UNIX based	NFS BASED

(source: Tanenbaum)

# Cohérence (1)

Sémantique	Commentaire
Atomique	Les opérations sur un fichier sont instantément visibles par tous les processus
Séquentielle	Aucun changement n'est visible jusqu'à ce que le fichier soit fermé
Fichiers immutables	Pas de mise à jour possible. Simplifie le partage et la réplication
Verrouillage	Le fichier est accessible en lecture ou en écriture par un seul processus
Transaction	Tous les changements sont atomiques

(Source: Tanenbaum)

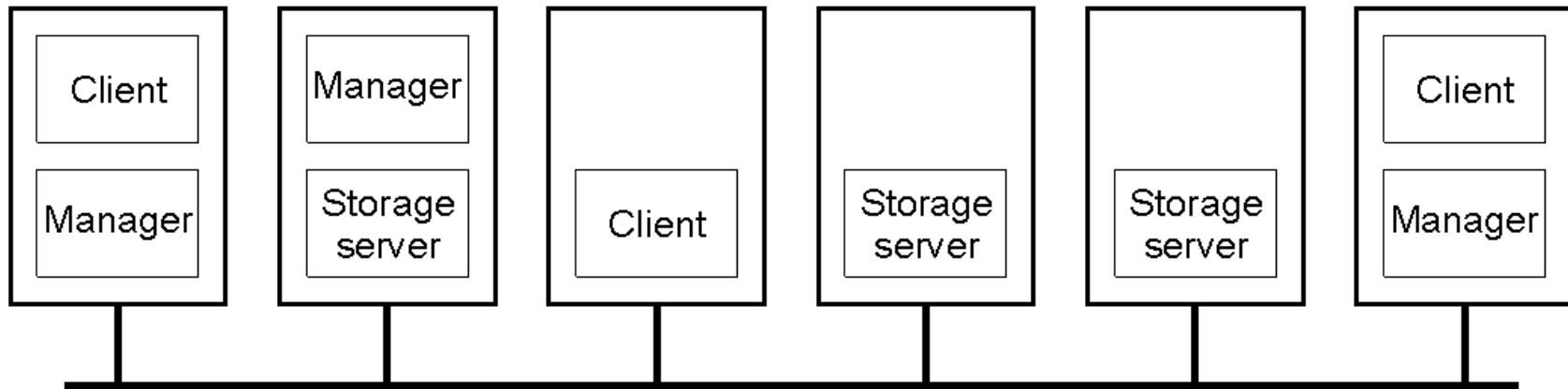
# Cohérence (2)

---

- Causale
- PRAM
- Cache
- Pour plus d'information, voir cours d'E. Gressier
- <http://deptinfo.cnam.fr/Enseignement/CycleSpecialisation/SAR/>

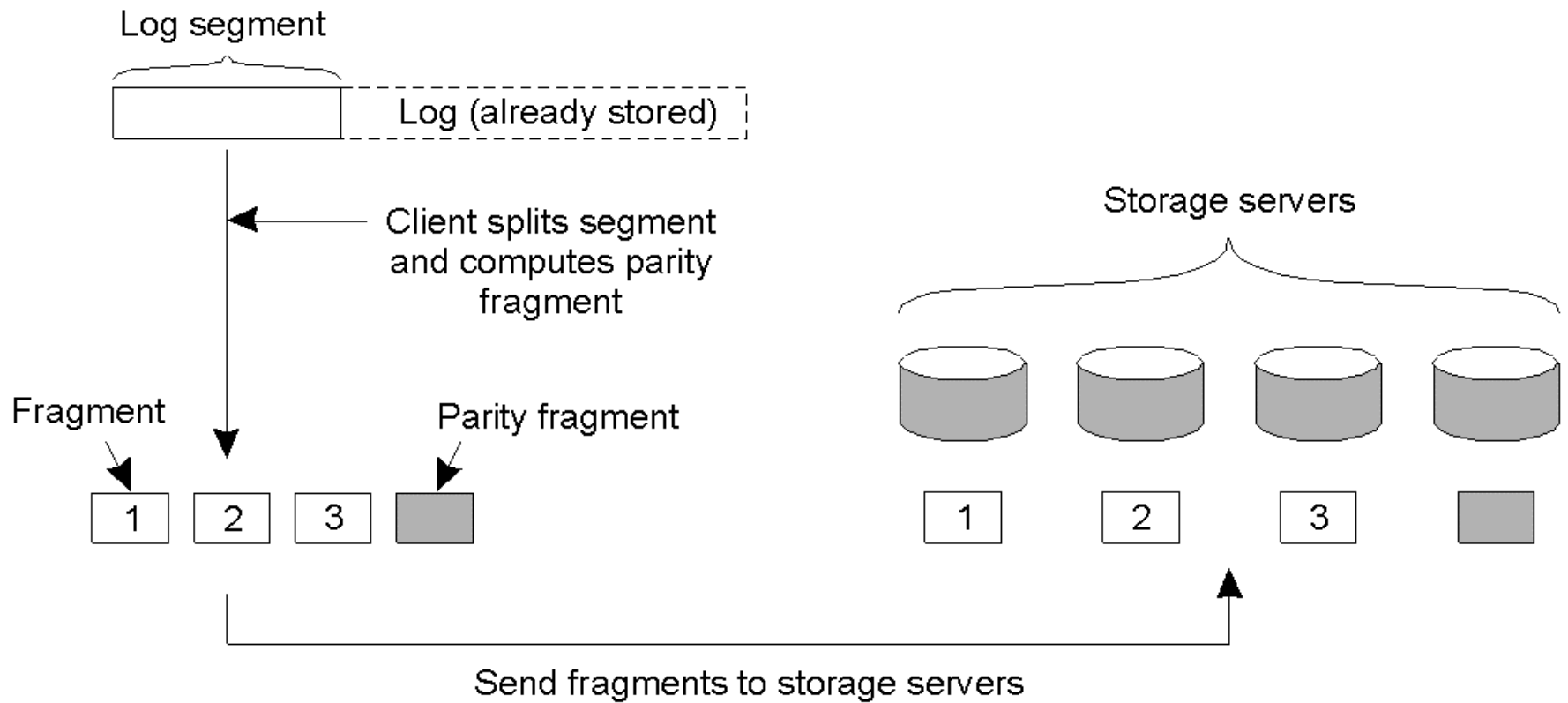
# xFS: Architecture

---



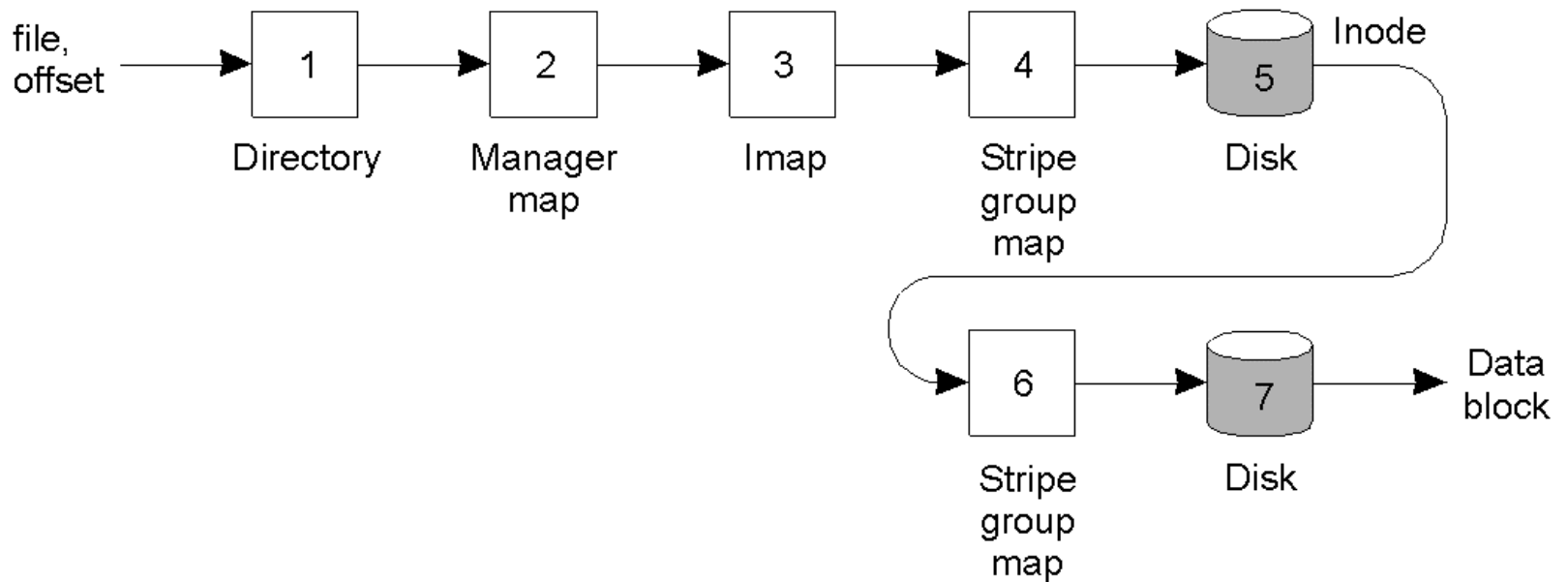
(source: Tanenbaum)

# xFS: principe du log based storage



(source: Tanenbaum)

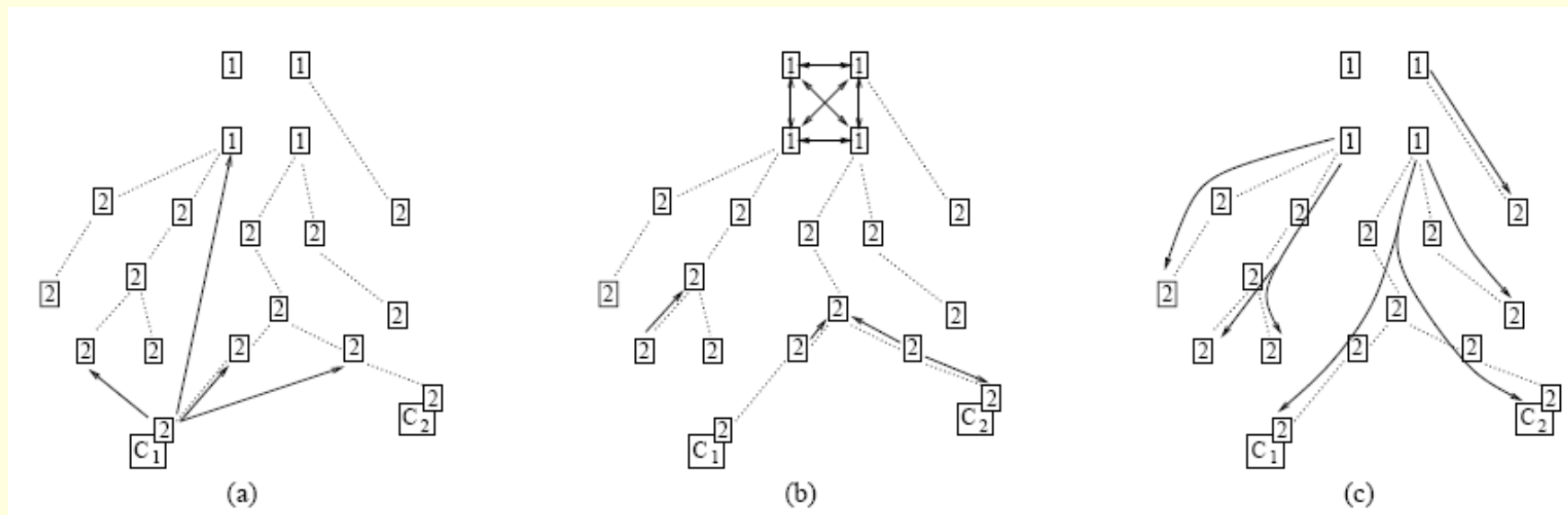
# xFS: une lecture



(source: Tanenbaum)



# OceanStore : Update

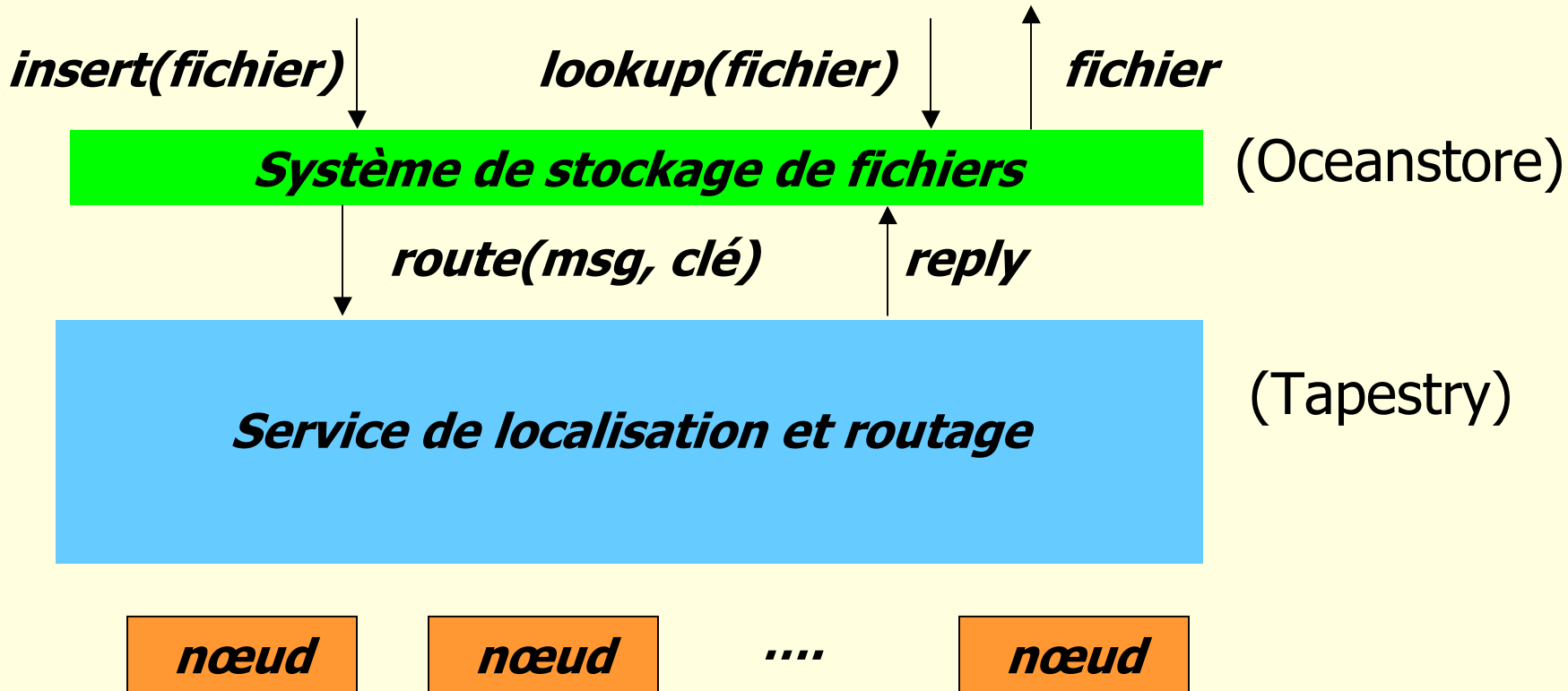


- Pas de synchronisation offerte : OceanStore compte sur la non-concurrence des écritures  
Si collision → Action annulée
- Mécanisme d'ordonnement : mise à jour sur un nombre restreint de pairs (pairs primaires)



# OceanStore :

## Tapestry : localisation et routage combinés



# S3: code de test

---

- <http://jroller.com/page/silvasoftinc/Weblog?catname=%2FjSh3ll>
- Shell écrit en Java basé sur le Shell d'Amazon

# S3: reporting de facturation

Service	Operation	UsageType	Resource	StartTime	EndTime	UsageValue
AmazonS3	GetService	Request		05/20/06 17:00:00	05/21/06 17:00:00	5
AmazonS3	GetObject	DataTransfer-Bytes	nicolas	05/20/06 17:00:00	05/21/06 17:00:00	10503994
AmazonS3	GetObject	Request	nicolas	05/20/06 17:00:00	05/21/06 17:00:00	3
AmazonS3	ListBucket	Request	nicolas	05/20/06 17:00:00	05/21/06 17:00:00	2
AmazonS3	ListBucket	DataTransfer-Bytes	nicolas	05/20/06 17:00:00	05/21/06 17:00:00	1122
AmazonS3	CreateBucket	Request	nicolas	05/20/06 17:00:00	05/21/06 17:00:00	1
AmazonS3	PutObject	Request	nicolas	05/20/06 17:00:00	05/21/06 17:00:00	4
AmazonS3	PutObject	DataTransfer-Bytes	nicolas	05/20/06 17:00:00	05/21/06 17:00:00	10538457
AmazonS3	GetService	DataTransfer-Bytes		05/20/06 17:00:00	05/21/06 17:00:00	1449

# S3: Exemple de requête SOAP

```
<PutObject xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>quotes</Bucket> <Key>Nelson</Key> <Metadata>
    <Name>Content-Type</Name> <Value>text/plain</Value>
  </Metadata> <Metadata> <Name>family</Name>
    <Value>Muntz</Value> </Metadata>
  <ContentLength>5</ContentLength> <AccessControlList> <Grant>
    <Grantee xsi:type="CanonicalUser">
      <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c6
        1be666e9</ID> <DisplayName>chriscustomer</DisplayName>
    </Grantee> <Permission>FULL_CONTROL</Permission> </Grant>
    <Grant> <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
    </Grantee> <Permission>READ</Permission> </Grant>
  </AccessControlList>
  <AWSAccessKeyId>1D9FVRAYCP1VJS767E02</AWSAccessKeyId>
  <Timestamp>2005-01-31T23:59:59.183Z</Timestamp>
  <Signature>luyz3d3P0aTou39dz bq7RrtSFmw=</Signature>
</PutObject>
```

# S3: Exemple de réponse SOAP

---

```
<PutObjectResponse
  xmlns="http://s3.amazonaws.com/doc/2006-
  03-01/"> <PutObjectResponse>
  <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9
  ac7f&quot;</ETag> <LastModified>1970-01-
  01T00:00:00.000Z</LastModified>
</PutObjectResponse>
</PutObjectResponse>
```

# Propriétés des services de stockage

	<i>Sharing</i>	<i>Persis- tence</i>	<i>Distributed cache/replicas</i>	<i>Consistency maintenance</i>	<i>Example</i>
Main memory	×	×	×	1	RAM
File system	×	✓	×	1	UNIX file system
Distributed file system	✓	✓	✓	✓	Sun NFS
Web	✓	✓	✓	×	Web server
Distributed shared memory	✓	×	✓	✓	Ivy (DSM, Ch. 18)
Remote objects (RMI/ORB)	✓	×	×	1	CORBA
Persistent object store	✓	✓	×	1	CORBA Persistent Object Service
Peer-to-peer storage system	✓	✓	✓	2	OceanStore (Ch. 10)

Types of consistency:

1: strict one-copy. 3: slightly weaker guarantees. 2: considerably weaker guarantees.