

## Séance 2

# Backup et restauration



Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

# Objectifs

- Grands principes du **backup et de l'archivage**  
*Exigences matérielles et stratégies*
- **Récupération de données** suites à un crash  
*Vérification des inconsistances et restauration*
- Système de fichiers **journalisé**  
*Principes des transactions et stockage de logs d'opérations*

# Backup et archivage



# Archivage

- Obligation légale de **garder informations** sur longue période

*Pression légale et compétitive sur les organisations*

- **Archivage** de données plus nécessaires immédiatement

- Stockage pas cher et lent, mais robuste et accès rapide
- Données gardées de manière fiable et authentique

- À ne pas confondre avec **backup** utilisé pour restauration

*Accès plus rapide, sauvegarde plus régulière...*

# Exigence matérielle (1)

- **Backup** et restauration en cas de désastre (DR)
  - Média de stockage de haute capacité
  - Grande performance pour lecture/écriture en streaming
  - Cout de stockage par Go bas
- **Archivage** de données et récupération éventuelle
  - Pouvoir assurer l'authenticité des données
  - Garantie d'une grande longévité étendue des données
  - Grande performance pour accès aléatoire en lecture
  - Cout total d'appartenance bas

# Exigence matérielle (1)

- Comparaison de **solutions matérielles** pour l'archivage  
*RAID, DVD, magnéto-optique, optique haute densité*

Attribut	RAID	DVD	MO	UDO
Véritable Write-Once	Non	Oui	Non	Oui
Longévité	Non	Oui	Oui	Oui
Removable	Non	Oui	Oui	Oui
Qualité professionnelle	Oui	Non	Oui	Oui
Capacité	Moyenne/Haute	Basse	Basse	Moyenne
Vitesse read/write	Haute	Basse	Moyenne	Moyenne
Vitesse access/seek	Haute	Basse	Moyenne	Moyenne
Cout total d'appartenance	Haut	Bas	Moyen/Haut	Bas

# World Backup Day

LE 1ER AVRIL, CERTAINES BLAGUES NE SONT PAS DRÔLES.

Soyez préparés. Sauvegardez vos fichiers le **31 mars**.

UNE SAUVEGARDE, C'EST QUOI ? ↴

PRENDRE L'ENGAGEMENT →

# Backup

- Résister aux **crashes des disques** magnétiques

*Backup pour ne pas perdre ces données pour toujours*

- Stockage des données sur un **autre dispositif**

*Dispositif physiquement différent et séparé du système*

- Utiliser **métadonnées des fichiers** pour backup intelligent

*Comparer date de dernier backup et de dernière modification...*

# Schedule de backup

- Un **cycle typique de backups** peut se dérouler comme suit
  - $J_1$  : Backup complet de tous les fichiers
  - $J_2$  : Backup incrémental des fichiers modifiés depuis  $J_1$
  - $J_3$  : Backup incrémental des fichiers modifiés depuis  $J_2$
  - ...
  - $J_N$  : Backup incrémental des fichiers modifiés depuis  $J_{N-1}$
- Après  $J_N$ , un **nouveau cycle** recommence en  $J_1$   
*Écraser les anciens backups ou nouveau média de stockage*

# Restauration

- Restauration des données à partir du full backup

*Suivi du rejeu de tous les backups incrémentaux*

- Possibilité de récupérer un fichier supprimé

*En remontant dans le backup précédent celui de la suppression*

- Longueur  $N$  du cycle de backups très importante

- Influence le nombre de médias de stockage à utiliser
- Définit le nombre de jours couverts par une restauration

# Stockage pour backup

- Tout **système de backup** dépend d'un système de stockage

*Pouvoir restaurer les données en cas de perte ou crash*

- **Trois caractéristiques** du stockage à utiliser

- Modèle du dépôt de données
- Média de stockage utilisé pour stocker les données
- Gestion du dépôt de données

# Modèle du dépôt

- Organisation et choix des **données stockées**

*Déstructuré avec inventaire papier et plusieurs médias stockage*

- **Cinq modèles** principaux utilisés en pratique

- Backup ou image système complète (pour configuration...)
- Incrémental stocke changement depuis backup précédent
- Différentiel stocke différences depuis dernier backup complet
- Delta inversé stocke différences pour états précédents
- Protection continue par sauvegarde de blocks diffs

# Média de stockage

- Utilisation d'un ou plusieurs **médias de stockage**

*Peut être plus ou moins efficace selon le modèle utilisé*

- **Plusieurs familles** de médias de stockage

- Bande magnétique grande capacité et peu chère
- Disque dur magnétique ou de type SSD pour la robustesse
- Stockage optique plus lent et plutôt pour archives (WORM)
- Service de backups à distance sur datacentres

# Gestion du dépôt

- Équilibre à trouver entre **accessibilité, sécurité et cout**

*Combinaison possible de plusieurs stratégies*

- **Cinq stratégies** principales et non exclusives

- Stockage on-line : disque interne ou array de type SAN
- Stockage near-line : jukebox avec mécanisme physique
- Stockage off-line : action humaine pour récupérer les médias
- Envoi backups dans bunker protégé (hardened, haute sécurité...)
- DR centre (disaster recovery) : backups et configuration...

# Récupération des données



# Crash et récupération

- **Fichiers et répertoires** stockés en mémoire principale et disque
  - Le système de fichiers doit résister à un crash du système
  - Risque de perte de données ou d'inconsistance
- **Deux attitudes** à avoir dans un système d'exploitation
  - Comment se protéger des crashes ou arrêts inopinés du système
  - Comment restaurer les données en cas de crash

# Backup physique et logique

- **Backup physique** sauvegarde les blocs disque
  - Backup de type incrémental des blocs
  - Récupération depuis système de backup lorsque corruption
- **Backup logique** des fichiers du système
  - Backup sélectif des données à protéger
  - Vérification des structures logiques lors du backup
  - Récupération de fichiers à restaurer

# Inconsistance de données

- Crash peut provoquer **inconsistances système de fichiers**
  - Mise à jour pas atomique de toutes les structures de données
  - Difficultés avec les différentes caches d'amélioration des E/S
- Corruption des **structures de données** multiples
  - Structure de répertoire, pointeurs des blocs libres et FCB*
- Corruption du système de fichiers à cause de **bugs**
  - Implémentation des systèmes de fichiers, contrôleurs disque...*

# Check de consistance

- Détection et correction des erreurs par le système de fichiers
  - Scan complet de toutes les métadonnées des fichiers
  - Bit de statut indiquant changement de métadonnées en cours
- Exécution d'un **consistency checker** lors d'une suspicion
  - Par exemple, l'outil `fsck` sous Unix
  - Compare structure de répertoires avec blocs sur le disque
- Réparation dépend algorithmes allocation/gestion espace libre

# Panne sur blocs disque

- Un **bloc disque corrompu** peut affecter plusieurs structures  
*Possibilité de restauration selon l'élément touché*
- **Cinq structures** principales peuvent être touchées
  - Bloc de boot peut être remplacé par un utilitaire
  - Super bloc peut être reconstruit s'il duplication
  - Liste des blocs libres en parcourant blocs atteignables
  - Quid des inodes, blocs indirects et blocs de données ?

# Crash et persistence

- Système de fichiers garantit la **persistence des données**
  - Fichier est gardé tant que l'utilisateur ne l'a pas supprimé
  - Possibilité de récupération après suppression grâce au backup
- **Difficulté** de garantir la persistance
  - Un crash détruit le contenu de la mémoire
  - Plus de cache augmente performance, mais données perdues
  - Opération sur fichiers manipule plusieurs blocs

# Crash

- Qu'est-ce-qu'un **crash** ?
  - Peut être vu comme deux threads avant/après le crash
  - Souci car lecture et écriture du même état partagé
  - Retour dans le passé car perte de l'état volatile actuel
- **Deux attitudes** possibles en réaction à un crash
  - Tout balancer et repartir de zéro
  - Essayer de reconstruire les données/informations perdues

# Mise à jour

- Les **mises à jour** doivent être consistantes
  - Ancienne ou nouvelle données, mais pas de déchets
  - Rendre plusieurs mises à jour atomique
- **Écrire métadonnées** d'abord ne garantit pas la consistance
- Exemple de modification du fichier /u/pi/foo
  - 1 Traverser la structure de répertoire jusque /u/pi/
  - 2 Allouer un bloc de données
  - 3 Mettre à jour le pointeur vers le bloc dans l'inode ( !!! )
  - 4 Écrire les données dans le fichier foo

# Mise à jour consistante

- Utiliser un ordre suivant une **approche « bottom-up »**
  - Mettre à jour les blocs de données du fichier
  - Modifier l'inode représentant le fichier
  - Changer la structure de répertoire, etc.
- Précautions à prendre lorsque **utilisation de caches**
  - Réécrire les blocs de données sur le disque
  - Mettre à jour inode/structure rép/... puis écrire sur disque
- Mise à jour consistantes, mais il reste des **déchets potentiels**

Système de  
fichier journalisé



# Journalisation

- Utilisation du même principe de **journalisation** que sur les DB  
*Système de fichiers orienté-transactions basé sur des logs*
- Pas toujours possible de **réparer une inconsistance**  
*Prend temps system/clock et pas moyen même avec aide humain*
- Sauvegarde de tout **changement de métadonnée** dans un log
  - Un ensemble d'opérations pour une tâche est une transaction
  - Retour de l'appel système une fois les changements commités
  - Log joué sur le système de fichier avant suppression de l'entrée

# Transaction

- **Groupement d'opérations** pour garantir les propriétés ACID
  - **Atomicité** garantit que ça s'est produit ou non (pas de partiel)
  - **Consistance** car transformation correcte de l'état
  - **Isolation** avec transactions les unes après les autres
  - **Durabilité** car ce qui s'est produit reste produit
- **Trois primitives** exécutables avec les transactions
  - BeginTransaction marque le début d'une nouvelle transaction
  - Commit finit la transaction de manière normale
  - Rollback annule la transaction et restaure l'état

# Fichier de logs

- Le **fichier de logs** est un buffer circulaire

*Précaution pour ne pas réécrire sur des entrées pas traitées*

- Précaution à prendre pour le **stockage** du fichier de logs

- Section complètement séparée du système de fichiers
- Sur un autre disque spindle avec têtes séparées

- **Après un crash**, examen du fichier de log

- Jouer les transactions commitées et pas exécutées
- Rollback de l'éventuelle transaction abortée

# Gain de performances

- Système de fichiers journalisé **améliore les performances**  
*Mise à jour plus efficace des métadonnées des fichiers*
- Meilleure utilisation **des E/S séquentielles vs aléatoire**
  - Écriture synchrone séquentielles dans le fichier de logs
  - Jeu des logs provoque écritures asynchrones aléatoires
- Meilleures performances des **opérations orientées métadonnées**  
*Typiquement les créations et suppressions de fichiers*

# Crédits

- <https://www.flickr.com/photos/godog/317312073>
- <https://www.flickr.com/photos/iloasiapacific/9039904148>
- <https://www.flickr.com/photos/jmv0586/2411746030>