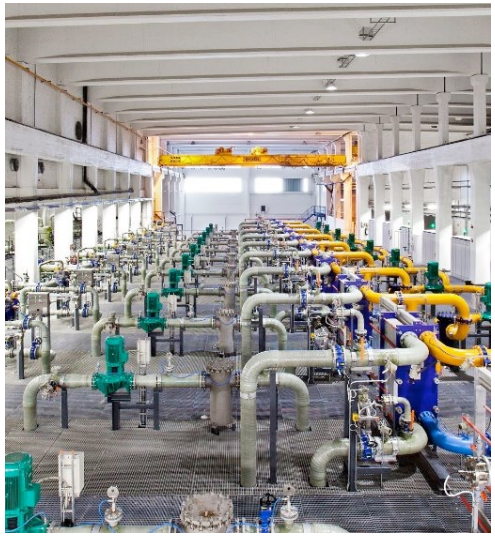
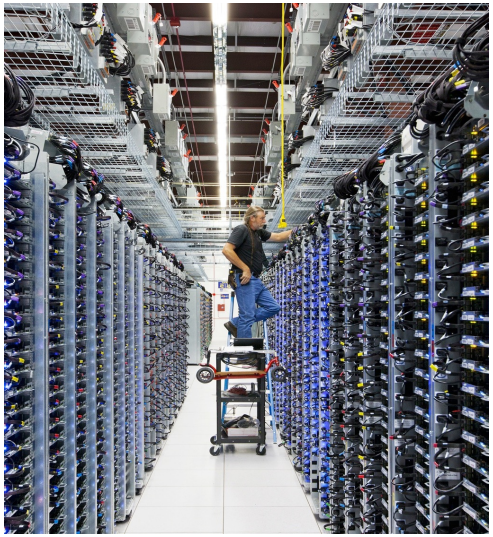
Administration IT — La sauvegarde


Année académique 2015/16

La sauvegarde

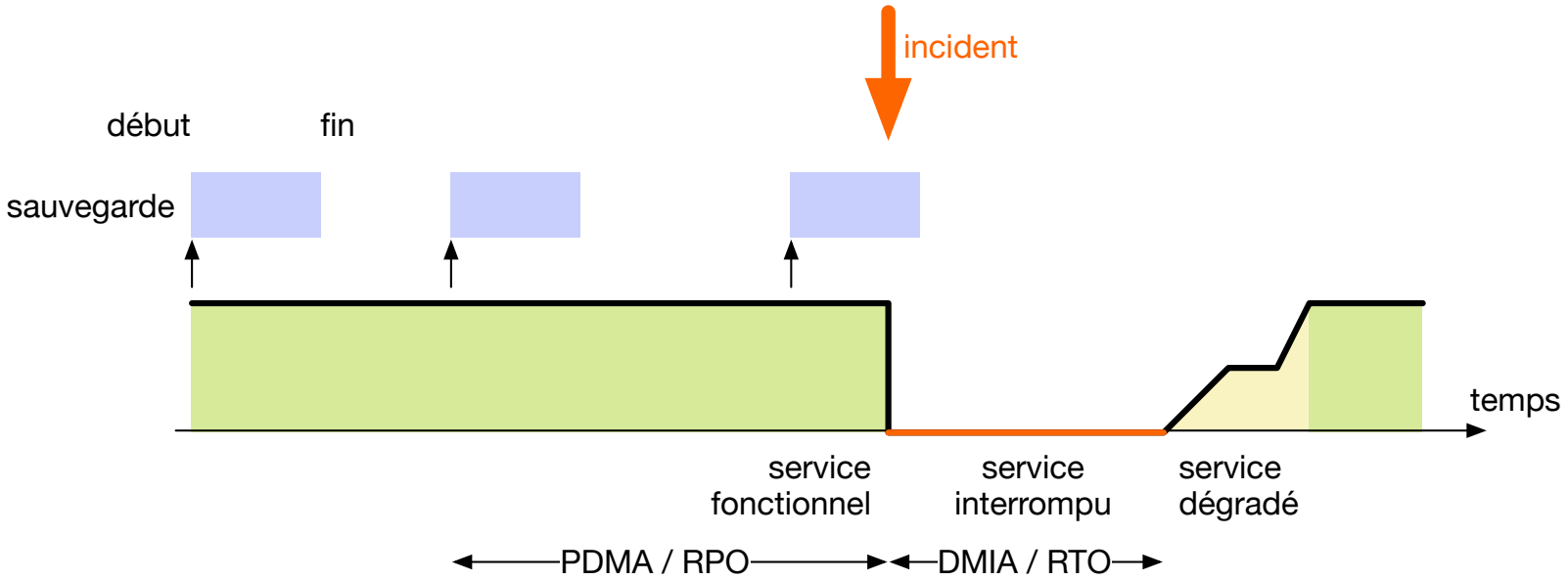
Introduction

- Chaque organisation IT doit développer un **plan de reprise d'activité** *(disaster recovery plan)*
  - La **sauvegarde** *(backup)* et la **restauration** *(restore)* font partie de ce plan
- Questions à poser :
  - Quelle est la compétence clé de l'organisation ?
  - Quelles sont les activités essentielles pour continuer la compétence clé ?
  - Quel système informatique supporte quelle activité ?
  - Pour chaque système informatique critique quels sont les paramètres RPO et RTO ?
  - Quels systèmes informatiques doivent être restaurés ensemble au même moment dans le passé (cohérence) ?
  - De quel type de sinistre chaque système doit-il être protégé ?
  - Quels seront les coûts d'une interruption de service ?
  - Quels seront les coûts d'une solution de sauvegarde et restauration ?

La sauvegarde

Objectifs

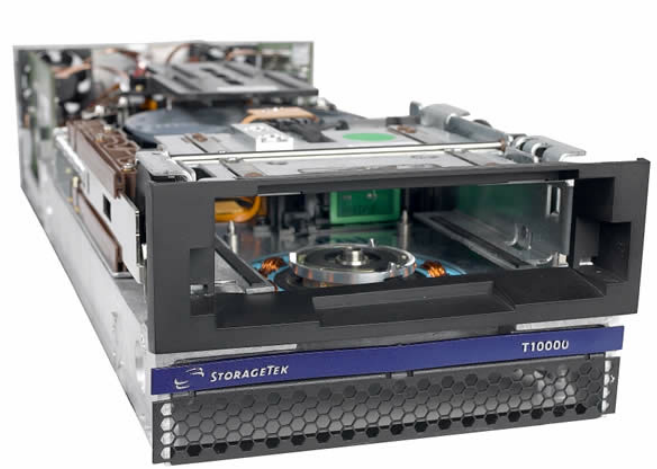
- Parmi les objectifs contenus dans le plan de reprise d'activité on trouve
  - La **perte de données maximale admissible** *(Recovery Point Objective, RPO)* : La quantité de données qu'un système IT peut être amené à perdre par suite d'un incident.
  - La **durée maximale d'interruption admissible** *(Recovery Time Objective, RTO)* : Le temps maximal acceptable durant lequel le service IT peut ne pas être fonctionnel après un incident.



La sauvegarde

Sauvegarde sur bande magnétique

- Les dernières générations de bande magnétique obtiennent une capacité de 185 TB par cartouche.



- Les lecteurs ont généralement une interface SCSI.

- Une librairie de sauvegarde peut contenir des milliers de cartouches.

La sauvegarde

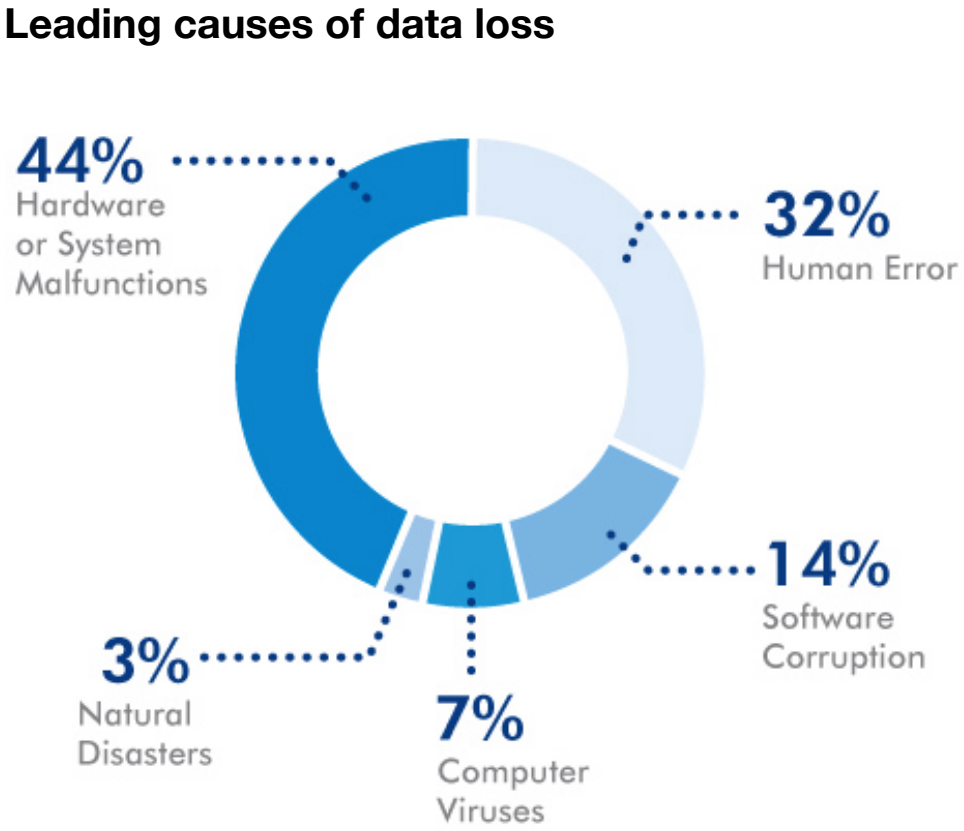
Classification des sinistres

- **Niveau 1** : Le sinistre affecte une application ou un serveur entiers
  - Défaillance de disque
  - Sabotage interne
  - Terrorisme informatique (attaque de dénis de service)
  - Attaque d'un employé mécontent
- **Niveau 2** : Le sinistre affecte un centre de données entier
  - Incendie d'un bâtiment ou inondation
  - Sinistres naturels (ouragan, tornade, tremblement de terre)
  - Clôture de bâtiment (fuite de gaz chloré)
  - Terrorisme physique (bombe)
  - Employé vraiment mécontent
  - Perte de toute connexion réseau
  - Perte de toue alimentation électrique
- **Niveau 3** : Le sinistre affecte un campus, une ville ou une zone métropolitaine
  - Sinistres naturels de grande envergure
  - Terrorisme physique (bombardement d'une centrale électrique)
  - Guerre

La sauvegarde

Protection contre la perte de données

- Parmi les raisons pour une perte de données on trouve :
  - **Erreur d'utilisateur** : "J'ai accidentellement écrasé mon rapport de laboratoire sur lequel j'ai travaillé pendant cinq heures !"
  - **Erreur du staff IT** : Un administrateur système supprime accidentellement le dossier personnel d'un utilisateur.
  - **Défaillance du matériel**
    - Défaillance de disque : Perte de secteurs individuels, crash de la tête de lecture
    - Défaillance de système : Défaillance d'une carte périphérique, de l'alimentation
  - **Défaillance du logiciel** : Bogues du système d'exploitation, de la base de données, de l'application
  - **Intrusion informatique, vandalisme, vol, racket** : Après une intrusion toutes les données qui étaient accessibles au malfaiteur sont suspectes
  - **Sinistres naturels** : ouragan, tornade, tremblement de terre, inondation
  - **Autres sinistres** : bombe



La sauvegarde

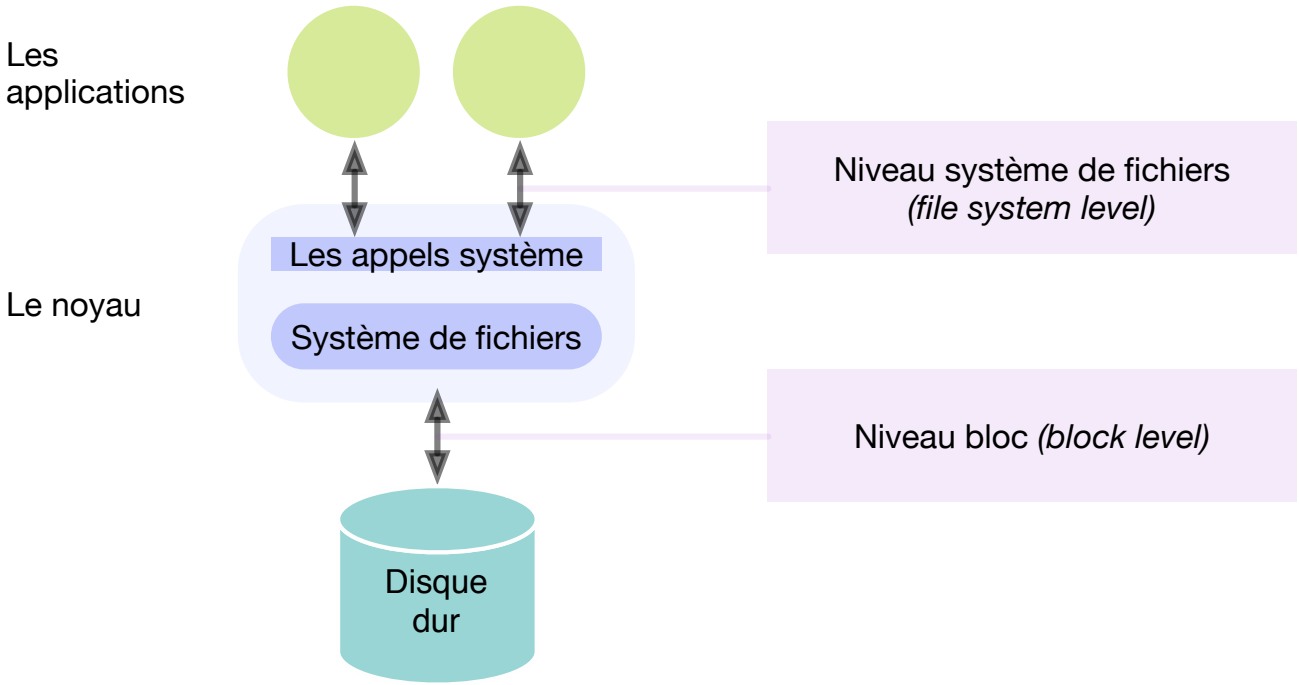
Sauvegarde contre archivage

- L'archivage a des points en commun avec la sauvegarde, mais une solution de sauvegarde n'est pas nécessairement une solution d'archivage
  - **Archivage** : Certains documents importants (souvent de nature légale) doivent être conservés plusieurs années, voire décennies.
  - C'est au-delà de la période de sauvegarde.
  - Pour ces documents il est nécessaire de développer un plan d'archivage.

Les systèmes de fichiers

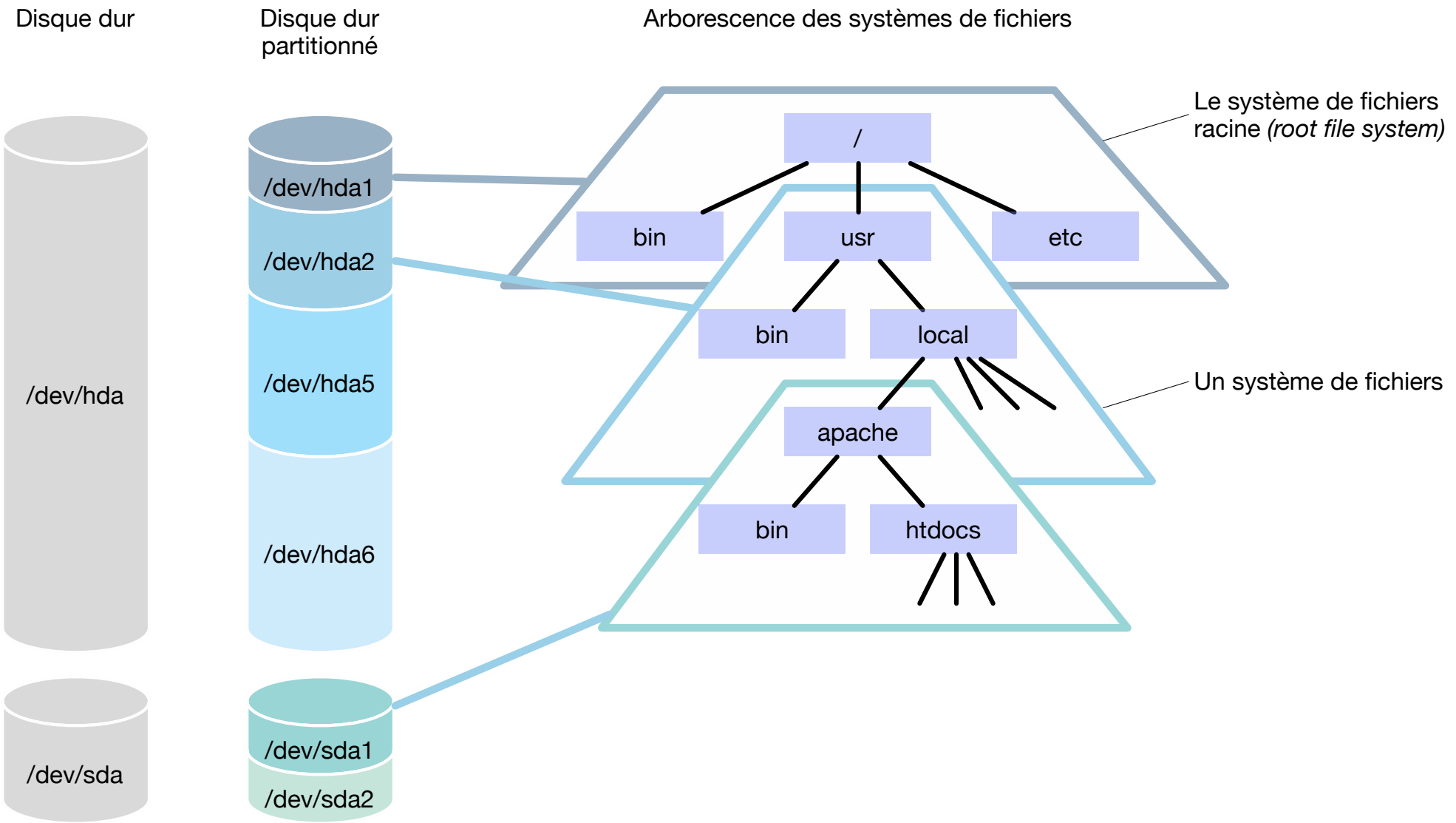
Rappel : Les niveaux bloc et système de fichiers du stockage

- Quand on parle du stockage on doit différencier entre deux niveaux d'abstraction :
  - Le niveau *système de fichiers*, utilisé par les applications
    - Fichiers
    - Répertoires
    - Métadonnées : Temps de modification, accès, permissions, ...
    - Liens physiques et symboliques
  - Le niveau *bloc*, fourni par les disques durs
    - Secteurs



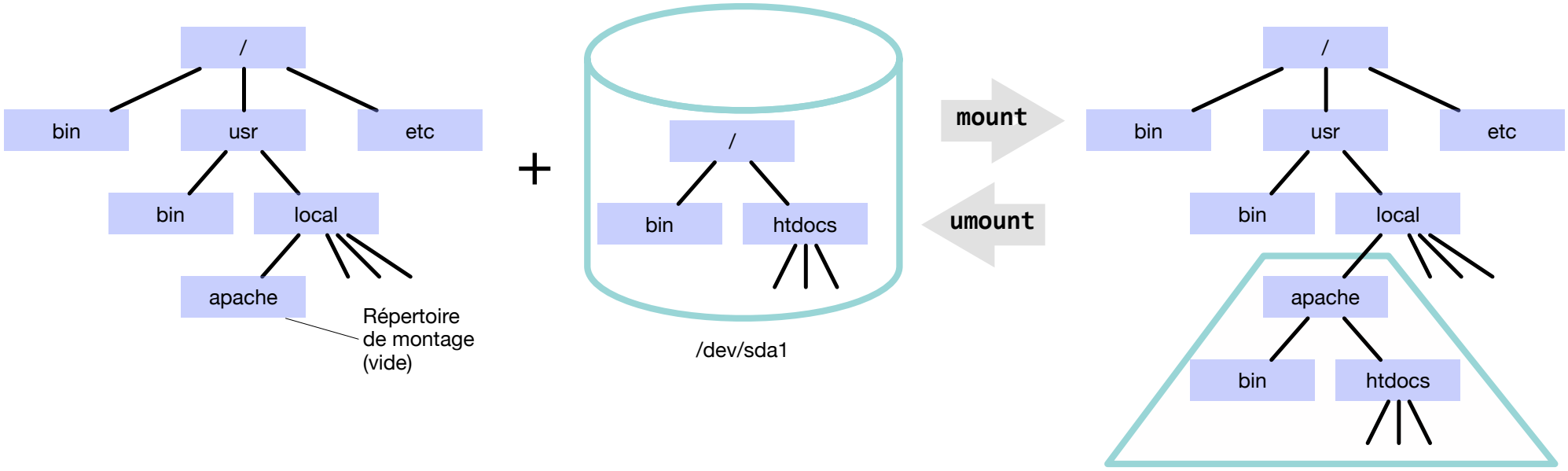
Les systèmes de fichiers

La relation entre disque durs, partitions et systèmes de fichiers



Les systèmes de fichiers

Monter et démonter un système de fichiers



Les systèmes de fichiers

Types

- Un système d'exploitation peut gérer différents types de systèmes de fichiers, chacun ayant des fonctionnalités différentes. Parmi les plus utilisés :
  - **ext2** — L'ancien FS standard Linux
  - **ext3** — Successeur de ext2, compatible avec son prédécesseur et plus tolérant aux pannes (fichier journal)
  - **ext4** — FS standard de Linux, successeur de ext3, support de volumes plus larges et minimisation de la fragmentation
  - **Btrfs** — Futur FS standard de Linux
  - **ISO 9660** — Le FS utilisé par les cdrom / dvdrom
  - **ReiserFS** — FS anciennement très répandu parmi les systèmes SUSE, premier FS Linux à incorporer un fichier journal
  - **XFS** — FS anciennement très répandu parmi les systèmes Debian
  - **ZFS** — FS avec fonctionnalités avancées développé originalement par Sun
  - **NTFS** — FS propriétaire de Microsoft, standard sur Windows et Windows Server
  - **FAT / VFAT** — FS standard sur DOS et anciennes version de Windows, utilisé souvent pour clés USB et dans l'informatique embarquée
  - **HFS+** — FS standard sur Mac OS X

Les systèmes de fichiers

Cycle de vie d'un système de fichiers sous Linux

- Connexion physique du disque dur au système : L'utilisateur attache le disque à un contrôleur de disque (ATA, SATA, SCSI, USB, ...).
- Détection du disque par le noyau : Le noyau reçoit un événement *hotplug* pour le nouveau disque sur le bus.
  - Il envoie un événement au système *udev* et met une description du disque dans le FS *sysfs* que *udev* peut récupérer.
- Création de fichiers spéciaux `/dev/xxx` représentant le disque : *udev* consulte les règles dans son répertoire de configuration `/etc/udev/rules.d`.
  - Il charge le pilote qui donne accès aux secteurs du disque si nécessaire.
  - Il crée plusieurs fichiers spéciaux dans le répertoire `/dev`.
    - Un pour le disque entier (/dev/hda) et un pour chaque partition (/dev/hda1, 2, ...).
- Montage du système de fichiers : *udev* appelle la commande `mount` pour monter les systèmes de fichiers contenus dans les partitions du disque.
  - `mount` consulte le fichier `/etc/fstab` pour déterminer le type de FS et l'endroit dans l'arborescence des répertoire où monter le FS.
  - `mount` charge le module noyau pour le type de FS si nécessaire.
- Le système de fichiers est monté dans l'arborescence et prêt à être utilisé par les applications.
- Signal de déconnecter le disque : L'utilisateur donne une commande pour déconnecter le disque dans l'interface graphique.
- Exécution de la commande `umount` : L'interface graphique appelle la commande `umount` pour chaque partition du disque.
  - `umount` ferme le système de fichiers (écriture des données dans le cache d'écriture sur disque) et l'enlève de l'arborescence.

Les systèmes de fichiers

Initialisation d'un disque sous Linux

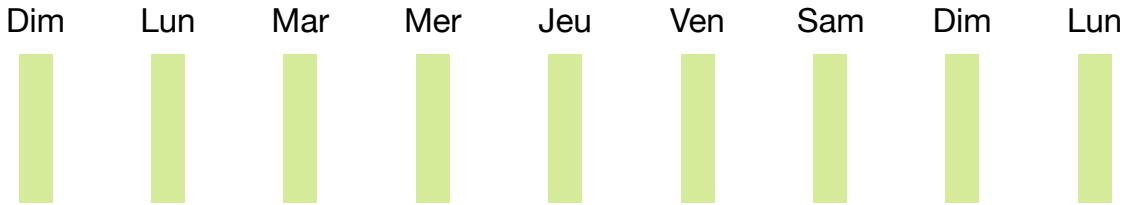
- Après avoir connecté un disque dur au système les étapes pour l'initialiser sont les suivantes :
  - Création d'une table de partition :
    - fdisk — Outil interactif de partitionnement des disques basique
    - sfdisk — Commande de partitionnement scriptable
    - parted — Outil interactif de partitionnement puissant
    - gparted — Version graphique de parted
  - Création d'un FS (formatage d'une partition) :
    - mkfs — Crée un FS du type donné
  - Montage / démontage d'un système de fichiers dans l'arborescence :
    - mount — Monte un FS
    - umount — Démonte un FS
  - Affichage d'informations sur les FS
    - df — Affiche les FS montés et pour chacun la place libre
  - Vérification / réparation d'un FS :
    - fsck — Vérifie / répare un FS. Le FS doit ne pas être monté.

La sauvegarde

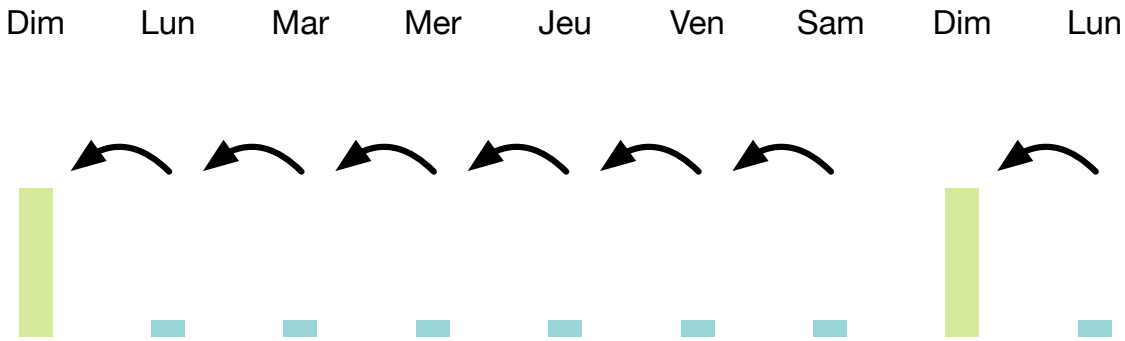
La sauvegarde incrémentale

Attention, l'industrie utilise ces termes pas toujours de manière cohérente !

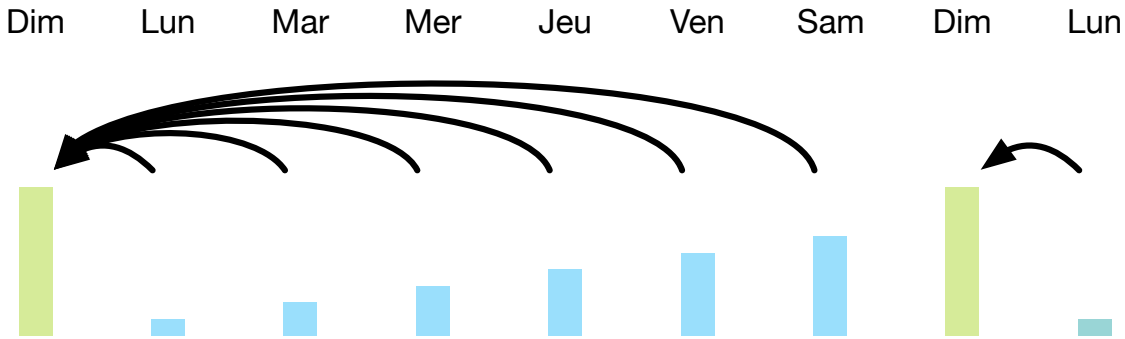
- Une **sauvegarde complète** *(full backup)* consiste à sauvegarder toutes les données.



- Une **sauvegarde incrémentale** *(incremental backup)* consiste à ne sauvegarder que les modifications effectuées depuis la dernière sauvegarde.



- Une **sauvegarde différentielle** *(differential backup)* consiste à ne sauvegarder que les données depuis la dernière sauvegarde complète.



La sauvegarde

Les niveaux de sauvegarde

- Les **niveaux de sauvegarde** *(backup levels)* permettent encore plus de flexibilité :
  - **Niveau 0** : Sauvegarde complète
  - **Niveau 1** : Une sauvegarde incrémentale qui contient tout ce qui a changé depuis la dernière sauvegarde complète. Si l'on répète des sauvegardes de niveau 1 on sauvegarde toujours tout depuis la dernière sauvegarde niveau 0.
  - **Niveaux 2-9** : Chaque niveau sauvegarde ce qui a changé depuis la dernière sauvegarde inférieure suivante. Par exemple le niveau 2 sauvegarde ce qui a changé depuis un niveau 1, ou un niveau 0 s'il n'y a pas de niveau 1.

La sauvegarde

Les cycles de sauvegarde

- Les sauvegardes peuvent être planifiées selon différents **cycles de sauvegarde** qui ont chacun leur avantages et inconvénients.

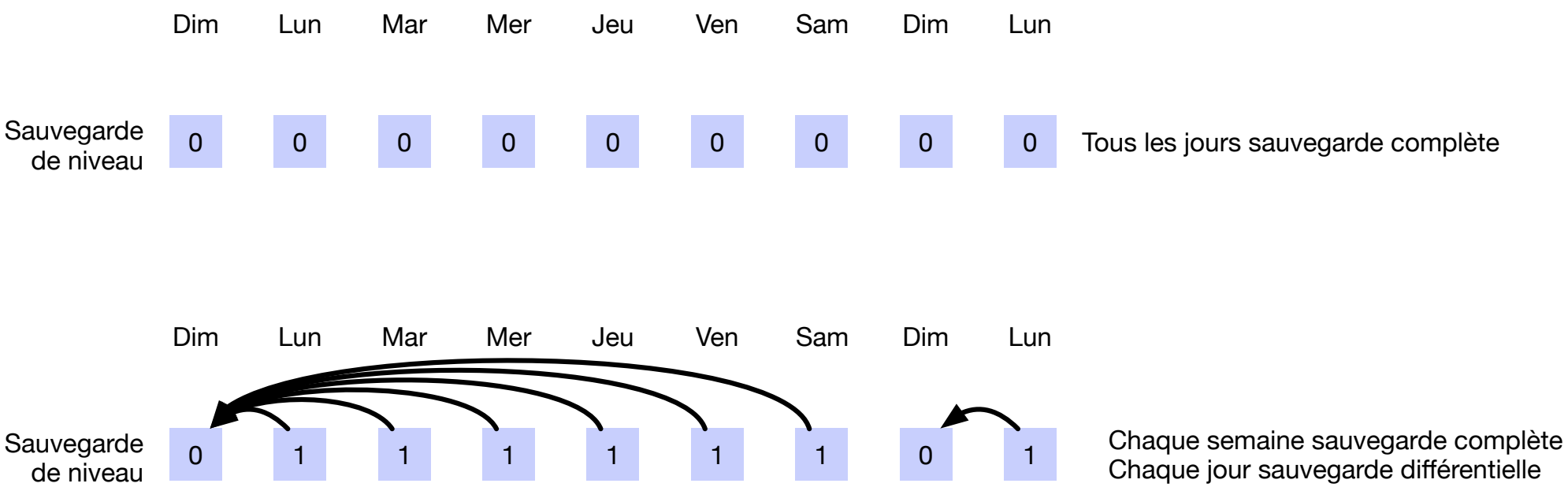
	Dim	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	Lun	
Sauvegarde de niveau	0	0	0	0	0	0	0	0	0	Tous les jours sauvegarde complète

	Dim	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	Lun	
Sauvegarde de niveau	0	1	1	1	1	1	1	0	1	Chaque semaine sauvegarde complète Chaque jour sauvegarde différentielle

La sauvegarde

Les cycles de sauvegarde

- Les sauvegardes peuvent être planifiées selon différents **cycles de sauvegarde** qui ont chacun leur avantages et inconvénients.



La sauvegarde

Les cycles de sauvegarde

	Dim	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	Lun	
Sauvegarde de niveau	0	1	2	3	4	5	6	0	1	Chaque semaine sauvegarde complète Chaque jour sauvegarde par niveau

	Dim	Lun	Mar	Mer	Jeu	Ven	Sam	Dim	Lun	
Sauvegarde de niveau	0	3	2	5	4	7	6	9	8	Chaque mois sauvegarde complète Chaque jour sauvegarde incrémentale type "Tours de Hanoi"

- Le **cycle de rétention** est le nombre de cycles de sauvegarde que l'on conserve.

La sauvegarde

Quelques commandes et outils de sauvegarde

	Linux	Mac OS X	Windows
Sauvegarde d'arborescences de fichiers	commandes tar, cpio, pax		Robocopy
		commande ditto	
Sauvegarde niveau bloc	commande dd		
Sauvegarde d'images de disque	outils Partimage, Clonezilla	outil Disk Utility	outil Ghost
Sauvegarde système incrémentale de FS	commandes dump/restore	outil Time Machine	outils NTBackup, Backup and Restore, File History
Sauvegarde complète (bare metal)	outils Mondo, Clonezilla		outil WBadmin
Sauvegarde client/serveur	outils Bacula, Amanda, BackupPC, ...		
Synchronisation de répertoires	commande rsync		

La sauvegarde

Unix — La commande **tar**

- La commande tar sauvegarde des fichiers ou arborescences de fichiers sur un fichier archive.
  - Le fichier archive peut être
    - Un fichier ordinaire
    - Un périphérique d'archivage local ou distant
    - La sortie standard
  - Le format de l'archive est compatible ISO. Les fichiers sont écrits les uns à la suite des autres, chacun précédé d'un en-tête. Celui-ci contient les métadonnées du fichier ainsi que son chemin.
- Le problème des chemins de fichiers
  - Une archive tar contient le chemin des fichiers sauvegardés. Le chemin est utilisé lors de la restauration pour placer les fichiers au bon endroit.
  - Normalement on veut des chemins relatifs. Ainsi on peut restaurer les fichiers à un autre endroit en changeant le répertoire courant. Les fichiers seront restaurés avec leurs chemins à partir du répertoire courant.
  - Particularité de la version GNU de tar : Si lors de la sauvegarde on indique un chemin absolu, le chemin complet des fichiers est sauvegardé, mais sans le premier / !
    - En réalité tar ne réalise que des sauvegardes relatives.
    - Il faut se placer à la racine au préalable, sinon les fichiers sont restaurés à partir du répertoire courant.

La sauvegarde

Unix — La commande **tar**

- Principales options
  - `-c` Sauvegarde
  - `-x` Restauration
  - `-t` Liste le contenu de l'archive
  - `-f` fichier Précise le chemin de l'archive
  - `-v` Afficher les fichiers actuellement traités
  - `-z` Compression (ou décompression) avec gzip
  - `-j` Compression (ou décompression) avec bzip2
  - `--exclude fichier` Exclut un fichier
- Utilisations courantes
  - Créer une archive à partir d'une arborescence : `tar -cf archive.tar repertoire_racine`
  - Lister le contenu d'une archive : `tar -tf archive.tar`
  - Restaurer une arborescence : `tar -xf archive.tar`
  - Travailler avec des archives compressées :

```
tar -czf archive.tar.gz repertoire_racine
tar -tzf archive.tar.gz
tar -xzf archive.tar.gz
```

La sauvegarde

La restauration *bare-metal*

- Principe
  - Une restauration *bare-metal* consiste à sauvegarder un serveur de telle manière que l'on puisse restaurer les données sur un nouvel ordinateur qui n'a aucun logiciel pré-installé *(bare-metal)*
- La sauvegarde *bare-metal* manuelle
  - On sauvegarde l'architecture du système : taille et usage des partitions (points de montage, options de montage, options de formatage des partitions...).
    # fdisk –l
    # cat /etc/fstab
  - On sauvegarde les FS systèmes (/, /usr, /var, /boot) avec les commandes incrémentales.
  - Remarque : la sauvegarde des autres FS (/home...) doit être régulièrement effectuée et ne fait pas stricto sensu partie de la sauvegarde *bare-metal*.
- La restauration manuelle
  - On démarre à partir d'un système bootable (CD-Rom...).
  - On restaure l'architecture du système (on repartitionne, on reformate).
  - On restaure les FS essentiels.
  - On rend le disque de démarrage bootable.
  - On redémarre le système.

La sauvegarde

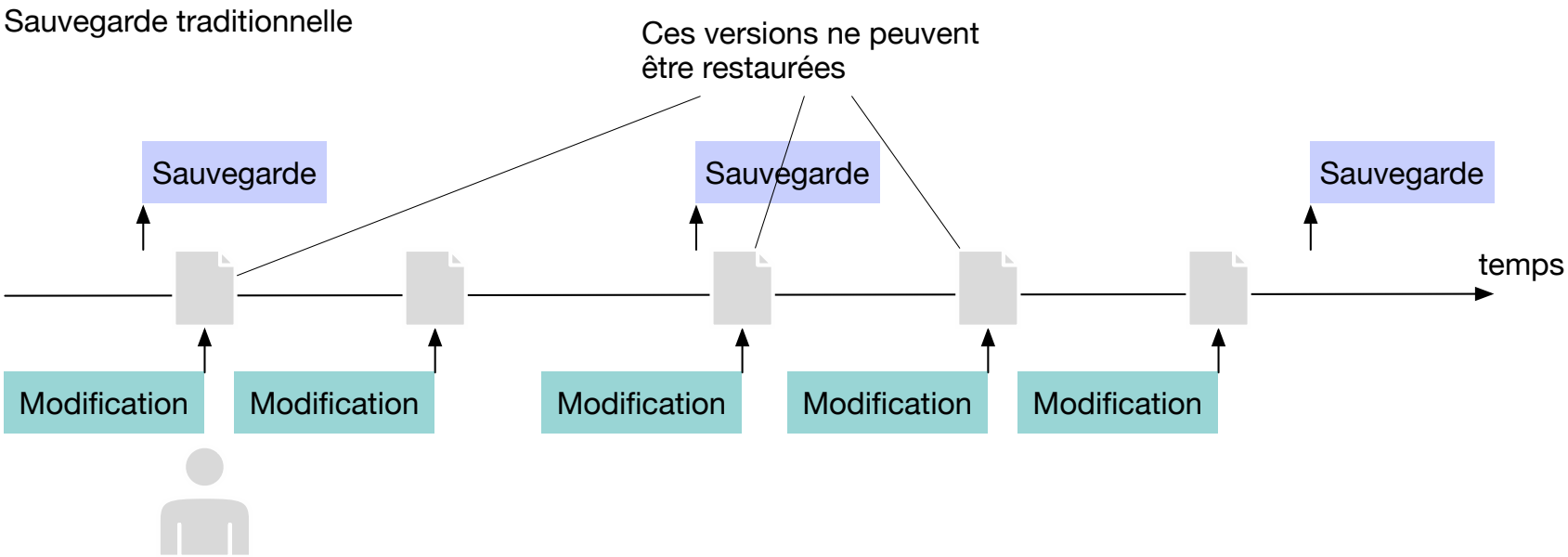
La restauration *bare-metal*

- Les commandes et les logiciels de sauvegarde *bare-metal*
  - dd — Il est simple de sauvegarder le disque système par cette commande.
  - Clonezilla — Produit réseau qui permet de cloner un système.
  - Mondo — Logiciel de sauvegarde *bare-metal*.
  - Bacula — Logiciel de sauvegarde client/serveur, simplifie la sauvegarde *bare-metal*.
  - Remarque : sur un système en RAID miroir, on peut prendre un des disques système, il peut servir de sauvegarde (il faut diminuer, voire arrêter l'exploitation).

La sauvegarde

Continuous Data Protection (CDP)

- Une sauvegarde traditionnelle ne permet pas de revenir à n'importe quelle version d'un fichier.
  - Les sauvegardes sont espacées (par exemple une sauvegarde par jour).
  - Mais il y a des fichiers qui sont mis à jour plusieurs fois dans la journée.
  - On ne peut restaurer que la dernière version d'un fichier avant la sauvegarde.

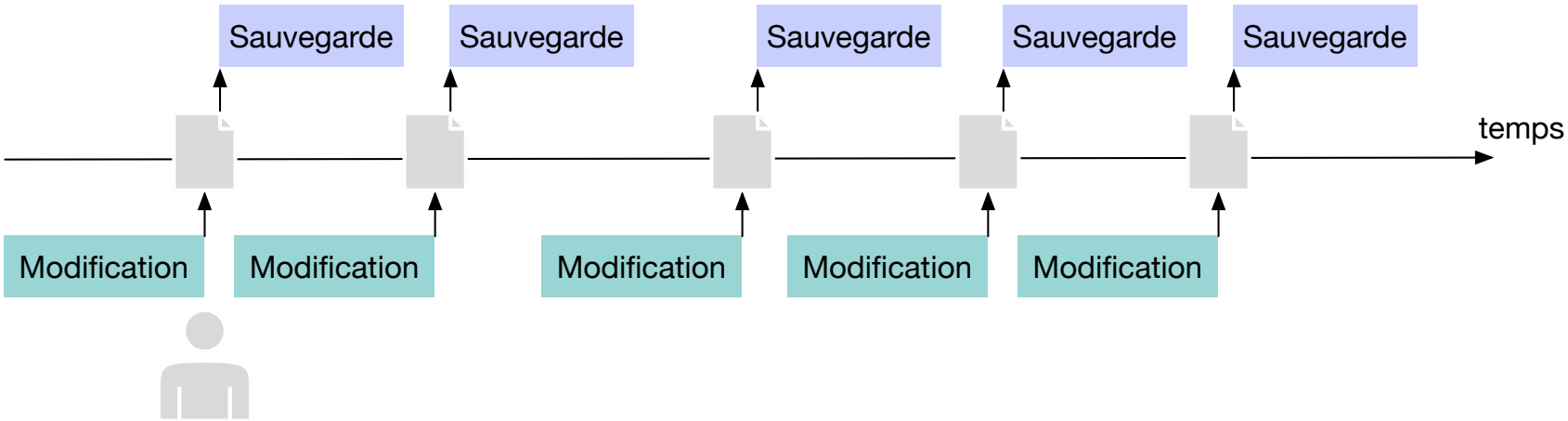


La sauvegarde

Continuous Data Protection (CDP)

- Une sauvegarde du type *Continuous Data Protection* crée une sauvegarde d'un fichier à chaque modification.
  - On peut restaurer le fichier à n'importe quelle version.
  - On obtient un RPO de zéro.

Sauvegarde *Continuous Data Protection*



La sauvegarde

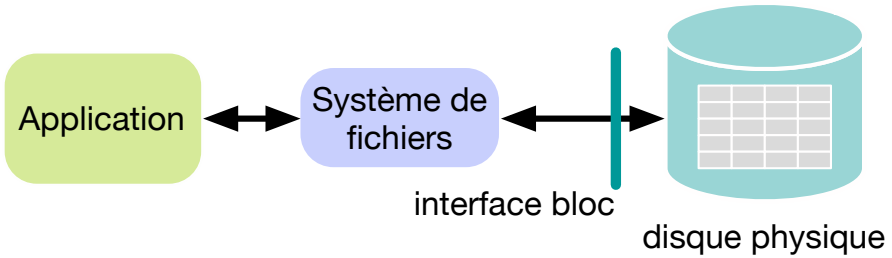
Near Continuous Data Protection

- Une sauvegarde du type CDP est coûteuse à mettre en oeuvre.
- Souvent on rencontre des systèmes du type *Near-Continuous Data Protection* qui font des sauvegardes toutes les *n* minutes.
- Ceci constitue une forte augmentation de la fréquence de sauvegarde comparé à l'approche traditionnelle (sauvegarde toutes les 24h). D'où certains problèmes avec une architecture traditionnelle :
  - La sauvegarde consomme beaucoup de ressources CPU, disque et réseau.
  - La restauration devient complexe avec un grand nombre de sauvegardes incrémentales.
- → Il faut des nouveaux mécanismes pour ces sauvegardes à haute fréquence.

Le stockage virtuel

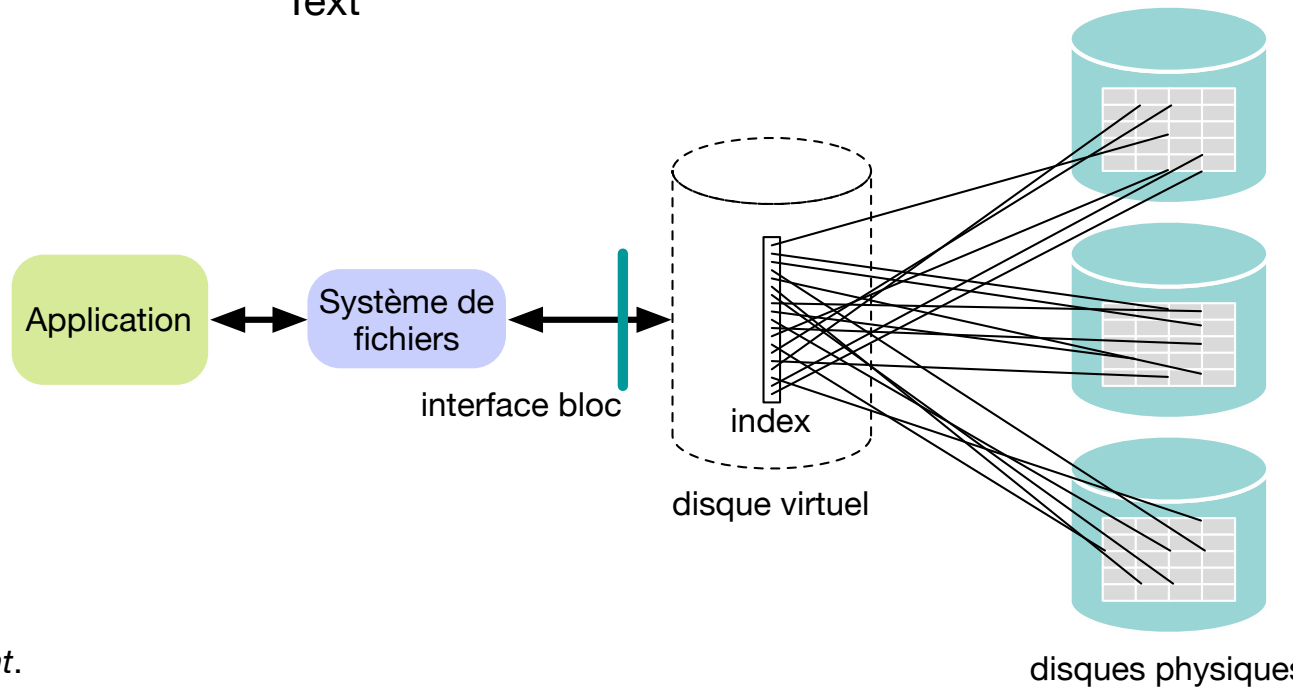
Principe

- Disque traditionnel
  - Il offre une interface très simple au système de fichiers :
    - Lire un bloc
    - Écrire un bloc



- Disque virtuel
  - Il présente la même interface au système de fichiers.
  - Les blocs sont remplacés par des pointeurs qui font partie d'un index. Chaque pointeur pointe vers un bloc* dans un disque physique.
  - Les disques physiques peuvent être locaux (p.ex. LVM) ou distants (SAN).

Text



* En réalité un groupe de blocs, appelé *extent*.

Le stockage virtuel

Motivation

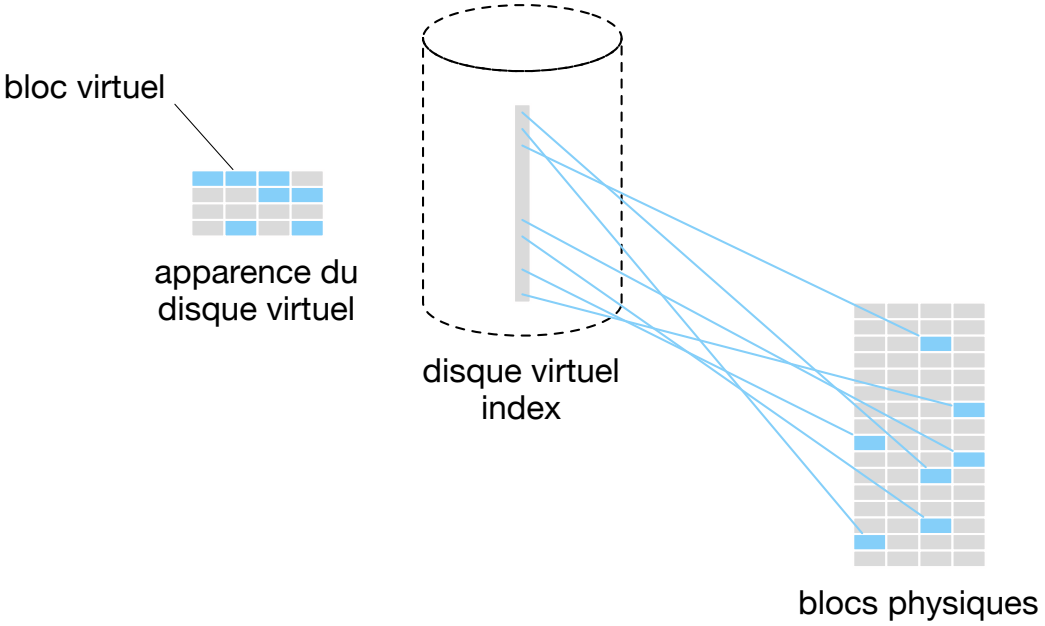
- Traditionnellement on trouve parmi les raisons pour utiliser du stockage virtuel :
  - La possibilité de créer des systèmes de fichiers à très grande capacité à partir de plusieurs disques physiques
  - La possibilité de changer dynamiquement la taille d'un disque virtuel et la capacité de son système de fichiers en ajoutant ou supprimant des blocs
  - En général une flexibilité plus grande pour la gestion des disques
  - (Accès à distance avec un SAN :) La possibilité de partager un pool de disques physiques entre plusieurs serveurs

Le stockage virtuel

L'instantané *(snapshot)*

- Une des fonctionnalités intéressantes du stockage virtuel est la possibilité de prendre des **instantanés** *(snapshot)* d'un disque virtuel.
  - Un instantané se comporte comme une copie complète du disque. Le processus de copie est quasi instantané.

Situation initiale : un disque virtuel

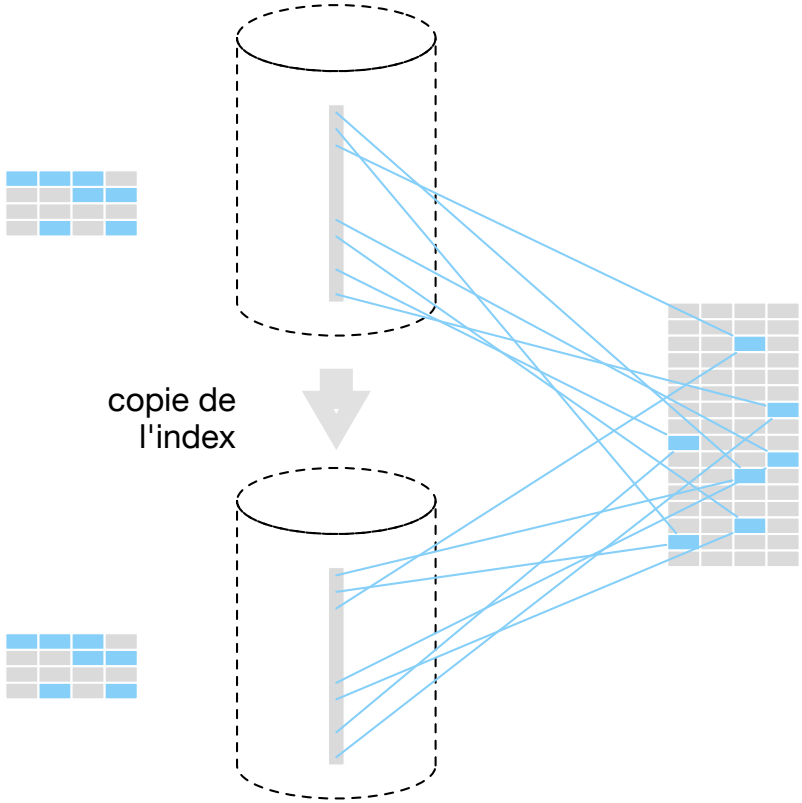


Le stockage virtuel

L'instantané *(snapshot)*

- Instantané : On crée un deuxième disque virtuel qui est une copie du premier.
  - Il suffit de copier l'index avec les pointeurs.
    - Opération très rapide
  - Le vrai travail s'effectue seulement après...

Snapshot : deuxième disque virtuel, copie du premier

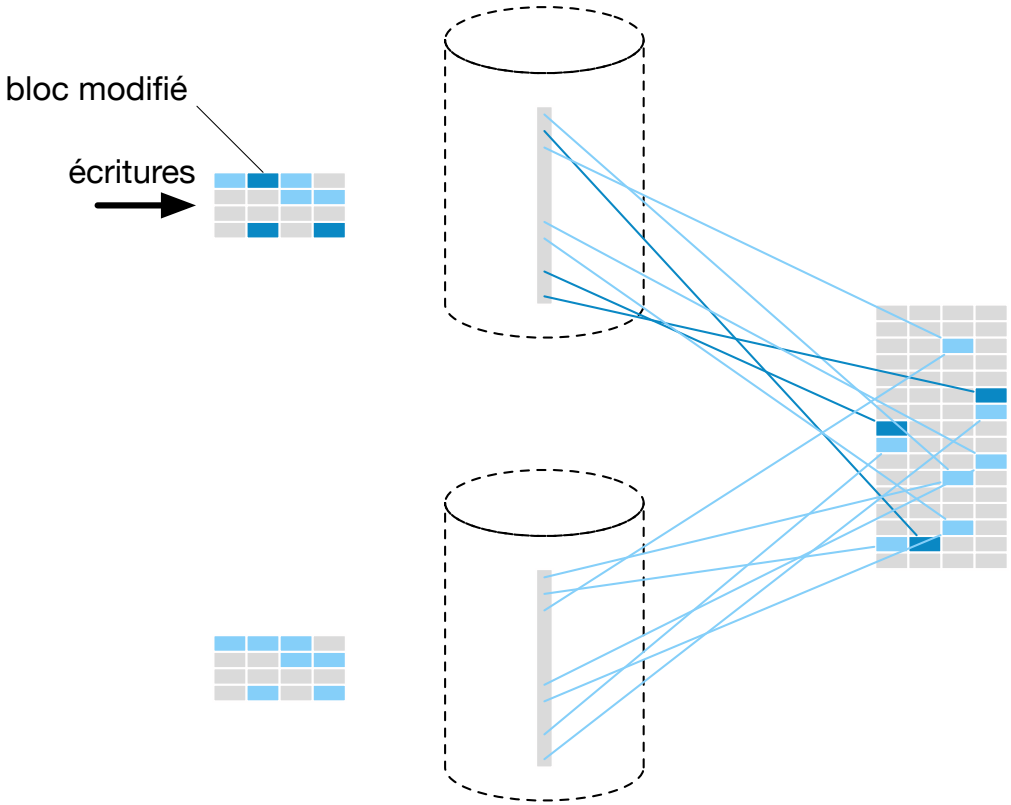


Le stockage virtuel

L'instantané *(snapshot)*

- Après avoir créé un instantané, les écritures sont traitées de manière spéciale. On procède selon le principe **copie à l'écriture** *(copy-on-write)* :
  - Avant de modifier le bloc physique on en fait une copie dans un nouveau bloc physique. On fait pointer le pointeur sur le nouveau bloc.
  - Ensuite on écrit dans le nouveau bloc.
  - Conséquences :
    - Les blocs qui ne sont pas modifiés sont partagés entre les deux disques.
    - Les modifications sur un disque virtuel n'affectent pas l'autre disque virtuel. Les deux disques sont totalement indépendants.

Après le snapshot : copie à l'écriture

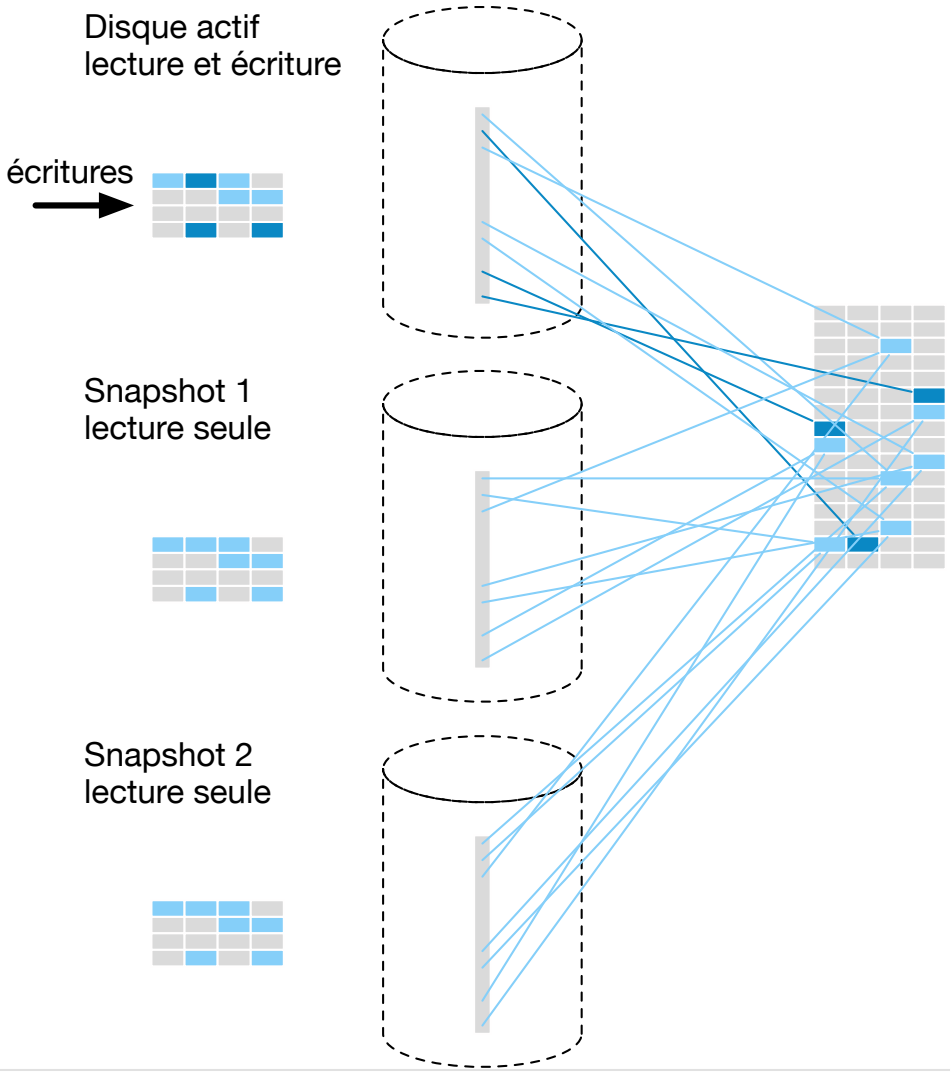


Le stockage virtuel

L'instantané *(snapshot)*

- Normalement on déclare un des disques virtuels le **disque actif** (accès en lecture et écriture) et on limite l'autre disque virtuel à la lecture seule (le **snapshot**).
  - Il reflète l'état du disque actif à l'instant quand on a fait le snapshot.
- On peut facilement multiplier les snapshots pour capturer l'état du disque à des instants différents.
- L'espace stockage total occupé par le disque actif et les snapshots dépend de la vitesse avec laquelle les données évoluent.

Plusieurs snapshots en lecture seule



Le stockage virtuel

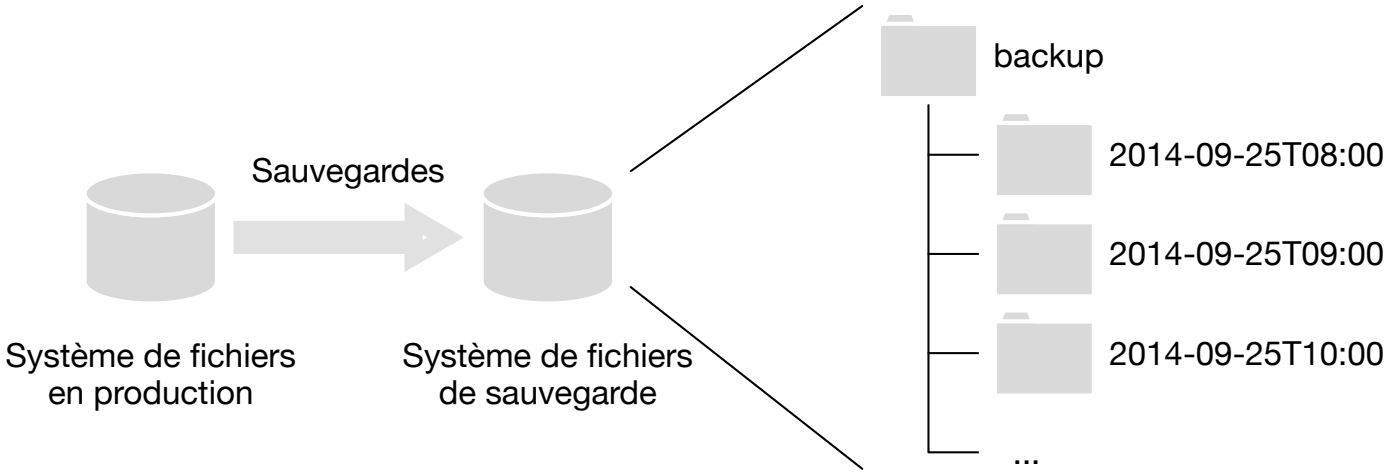
L'instantané *(snapshot)*

- La capacité de faire des instantanés se trouve souvent dans les systèmes de stockage qui ont une notion de disque virtuel, et réalisent une indirection grâce à une table de pointeurs.
  - Logical Volume Management
    - Linux Logical Volume Manager (LVM)
    - Windows Logical Disk Manager (LDM)
  - Storage Area Network (SAN)
  - La partie stockage des logiciels de virtualisation de plateforme
    - VMware
    - Hyper-V
    - VirtualBox
    - Parallels
    - ...

La sauvegarde

Les "snapshots" au niveau système de fichiers

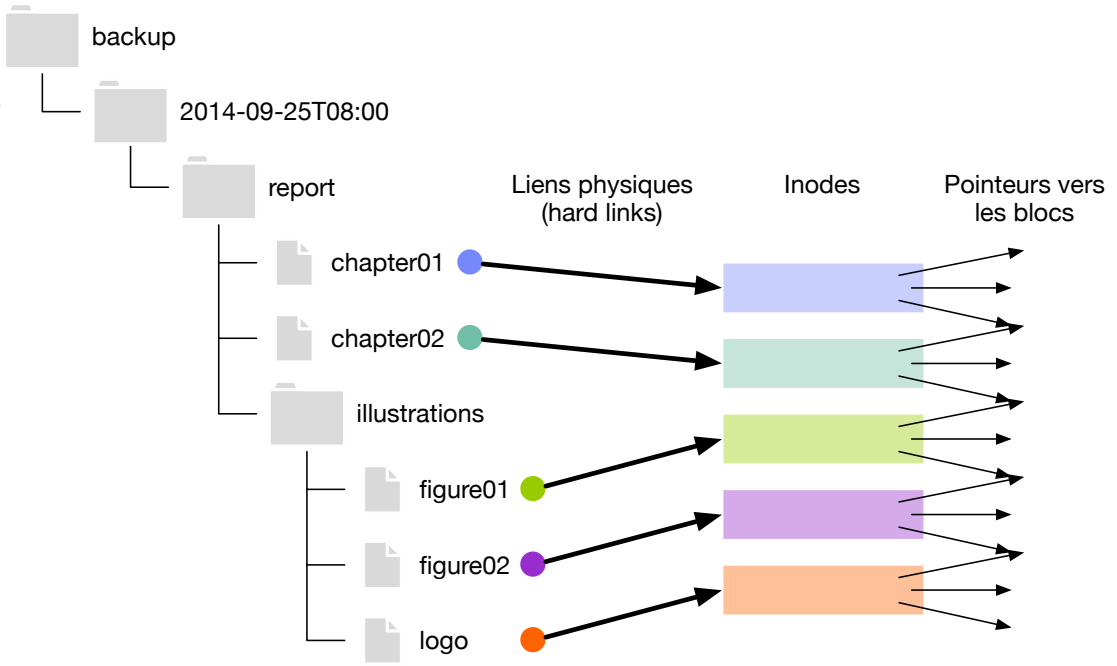
- Avec des sauvegardes incrémentales très fréquentes la restauration depuis archives devient onéreuse.
  - Il faut désarchiver toutes les archives dans la chaîne des dépendances.
- Une approche beaucoup plus facile à restaurer consiste à utiliser un disque avec un système de fichiers comme support de sauvegarde.
- Ce disque sera écrit lors de la sauvegarde, puis il reste en mode lecture seule.
- Sur le système de fichiers on utilise des liens physiques (hard links) comme pointeurs on peut réaliser des "snapshots" qui partagent les fichiers qui n'ont pas changés entre sauvegardes.



La sauvegarde

Les "snapshots" au niveau système de fichiers

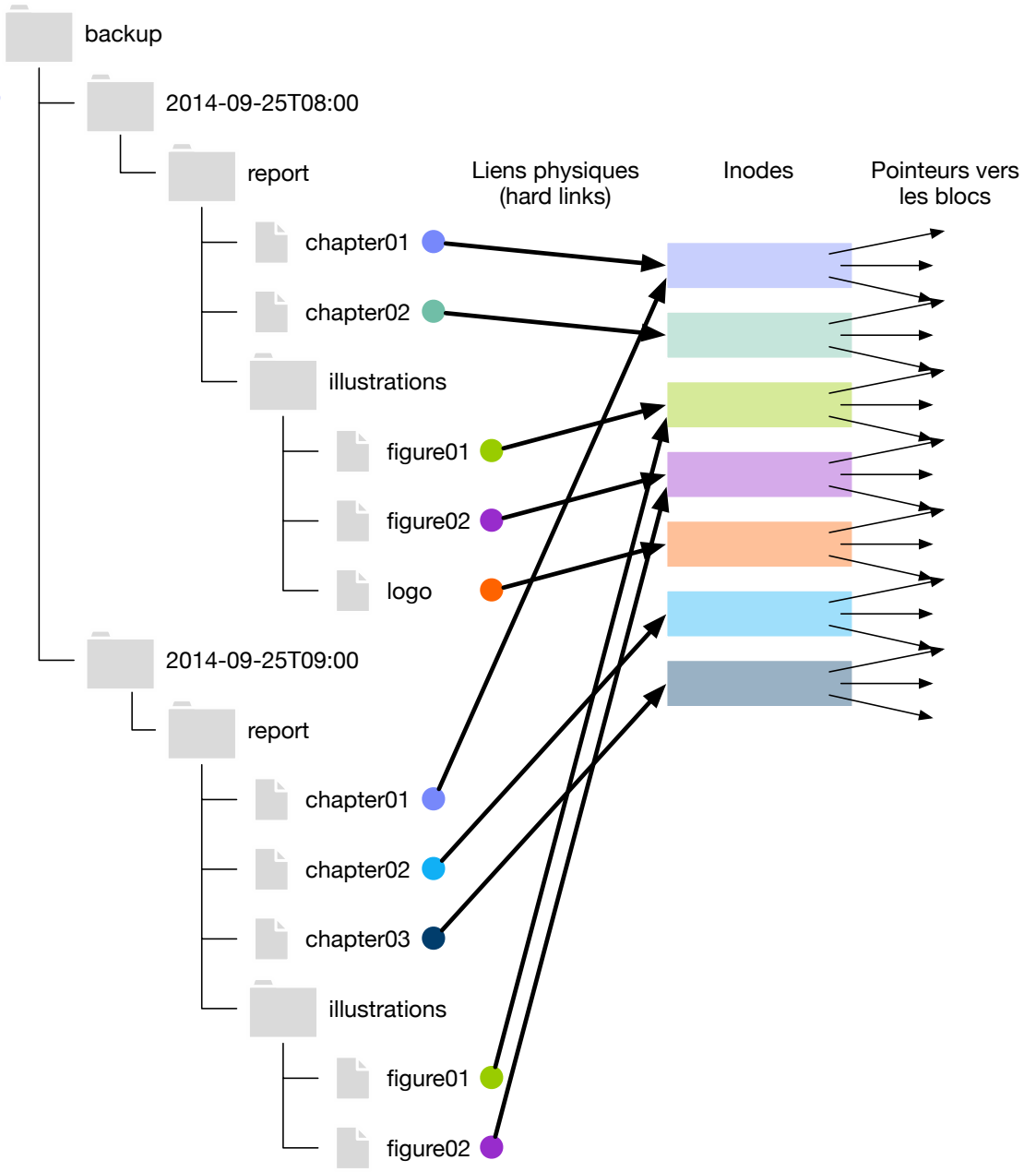
- Initialement on écrit une sauvegarde complète sur le disque : tous les fichiers sont copiés



La sauvegarde

Les "snapshots" au niveau système de fichiers

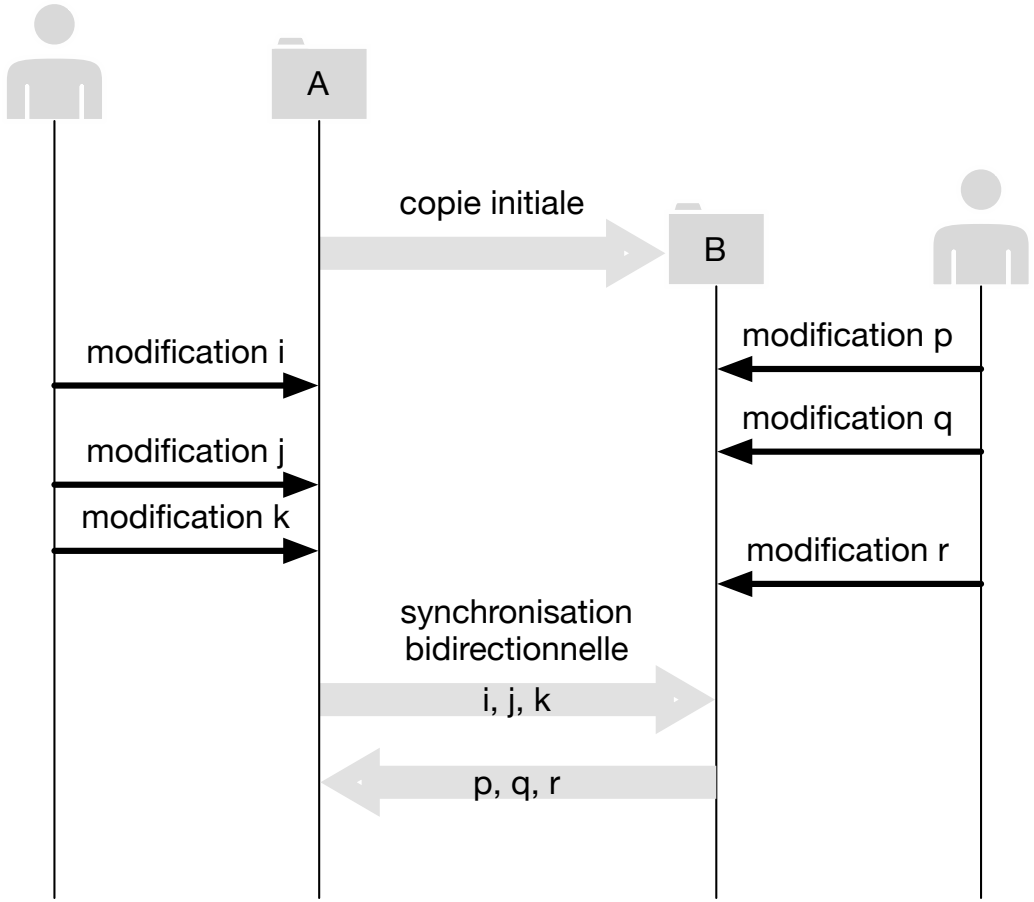
- La sauvegarde suivante sera incrémentale.
  - Un fichier qui n'a pas changé sera réalisé comme lien physique vers le fichier correspondant dans la sauvegarde antérieure.
  - Un fichier qui a changé sera copié comme nouveau fichier.



La synchronisation de fichiers

Introduction

- La synchronisation de fichiers consiste à faire correspondre les contenus de deux (ou plus) endroits de stockage.
  - On suppose que les contenus sont similaires.
  - On va donc faire quelque chose de plus intelligent et plus rapide qu'une copie complète : on transmet seulement les différences ou les changements depuis la dernière synchronisation.
- La synchronisation est avantageuse comparée à une copie complète quand les contenus évoluent lentement et la synchronisation doit être répétée. Exemples d'application :
  - Synchroniser les fichiers d'un PC, téléphone mobile ou tablette avec un stockage cloud : Dropbox, Google Drive, iCloud, ...
  - Synchroniser les contenus (répertoire www) d'un cluster de serveurs web.
  - Synchroniser les métadonnées des packages que nécessite un gestionnaire de packages.
  - Faire des sauvegardes.



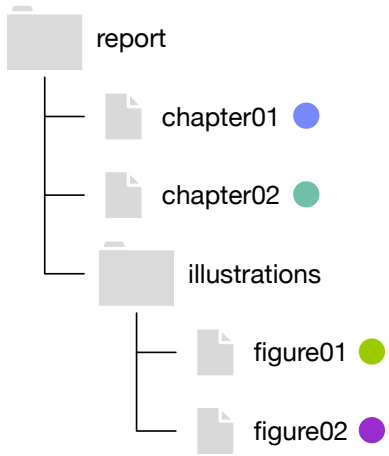
La synchronisation de fichiers

Introduction

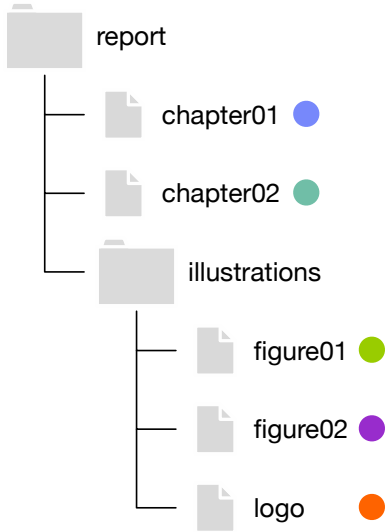
- Avec la synchronisation de fichiers on peut faire correspondre une arborescence de répertoires et fichiers A avec une arborescence B.
  - Si A et B sont sur la même machine, c'est une synchronisation locale.
  - S'ils sont sur deux machines différentes, c'est une synchronisation distante.
- Deux types de synchronisation :
  - Synchronisation unidirectionnelle A → B
    - Une des arborescences est déclarée la référence (A)
    - L'autre arborescence est modifiée pour lui correspondre (B).
    - Si B contient des modifications, elles seront perdues.
  - Synchronisation bidirectionnelle A ↔ B avec ancêtre commun
    - On suppose que A et B sont des modifications d'une même version originale qui est déclarée la référence (R)
    - Les modifications de A par rapport à R sont envoyés à B.
    - Les modifications de B par rapport à R sont envoyés à A.
    - Il peut y avoir des conflits logiques si le même objet a été modifié dans A et dans B.

La synchronisation de fichiers

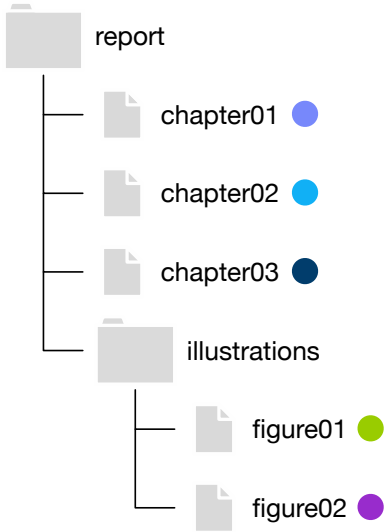
Exemple



A



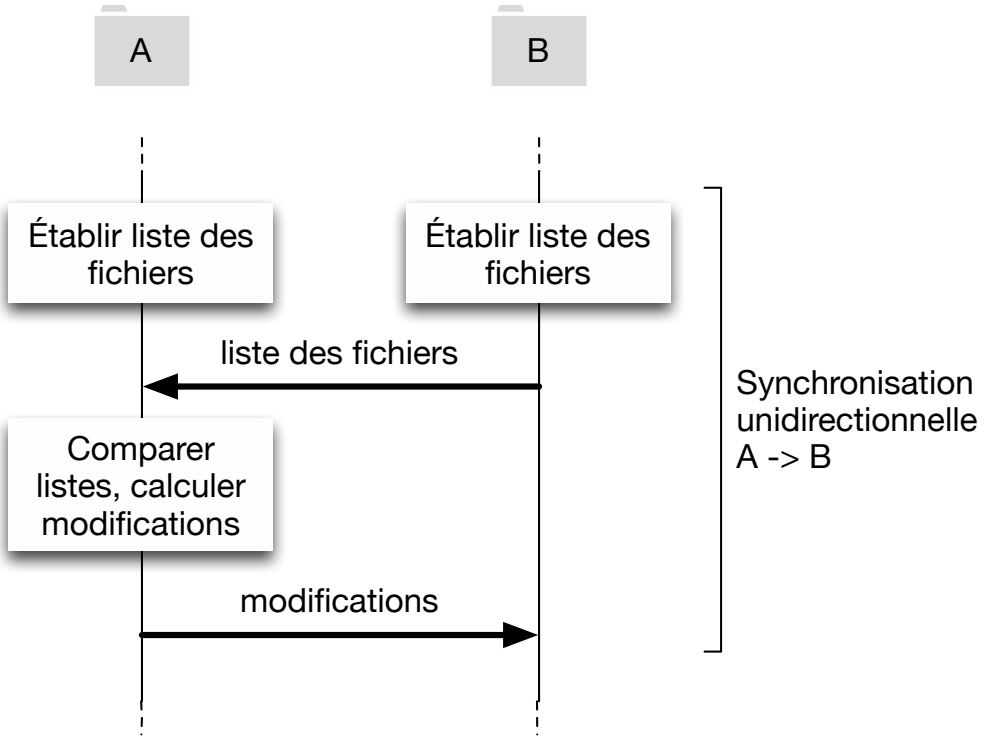
B



La synchronisation de fichiers

Principe

- La synchronisation peut être basée sur une de deux méthodes :
  - Sans historique : Comparaison de listes de fichiers
  - Avec historique : Capturer l'historique des modifications et les garder dans un journal
- La méthode de la comparaison des fichiers :
  - Chaque noeud A et B dresse la liste des fichiers et leurs métadonnées.
  - Le noeud qui va envoyer les modifications reçoit la liste de l'autre pour déterminer quelles modifications il faudra faire sur l'autre noeud. Ensuite il les envoie à l'autre noeud.
  - Modifications possibles :
    - Fichier ou répertoire ajouté
    - Fichier ou répertoire supprimé
    - Fichier ou répertoire modifié



La synchronisation de fichiers

L'outil **rsync**

- **rsync** est un outil de synchronisation de fichiers très puissant
- Logiciel Open Source (publié sous licence GPL), disponible pour systèmes Unix et Windows
- Inventé et développé par Andrew Tridgell et Paul Mackerras, première publication en 1996
- Utilise des algorithmes sophistiqués pour réduire le volume de données à échanger
- Peut faire usage de SSH pour établir une connexion sécurisée entre machines
- Inclut des options qui permettent de faire des sauvegardes différentielles
  - De nombreux outils de sauvegarde ont été créés sur la base de rsync

La synchronisation de fichiers

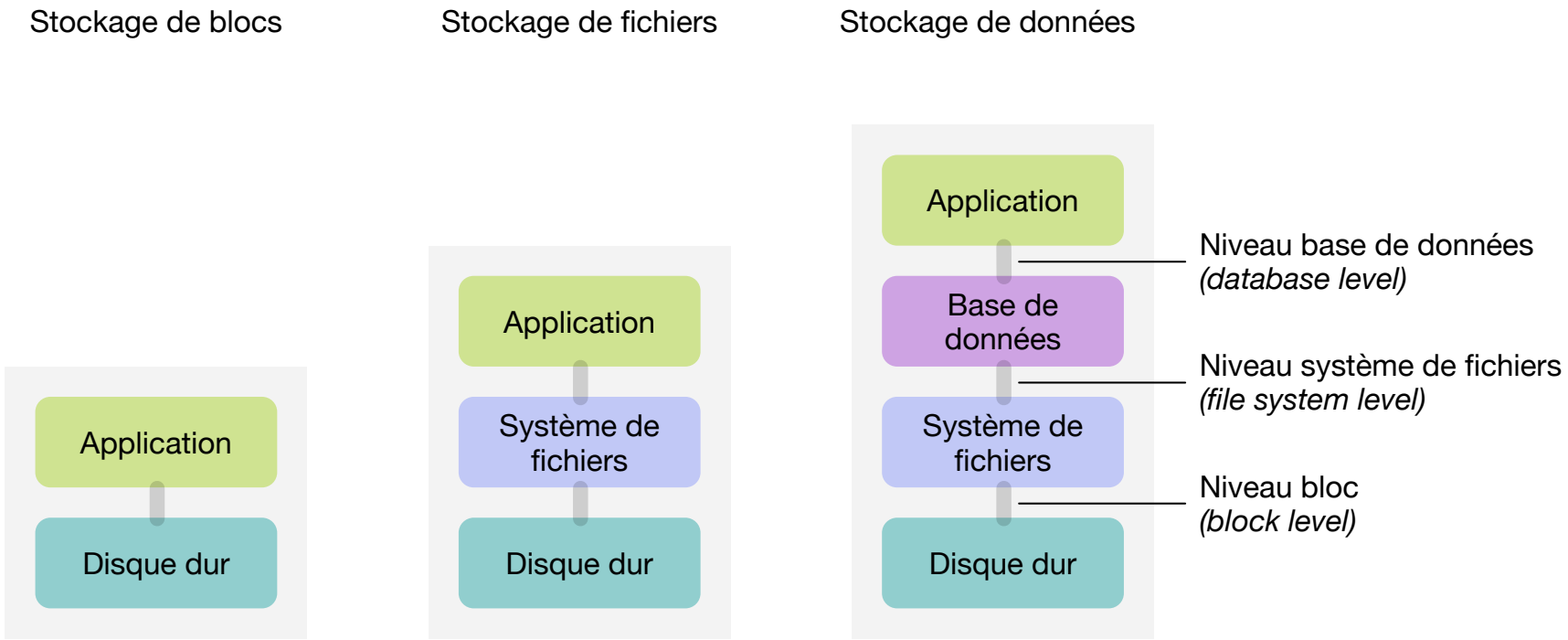
L'outil **rsync** — Fonctionnement

- Potentiellement il est très coûteux de déterminer si un fichier a un contenu différent sur A et B. Astuces utilisées par rsync :
  - On compare le fichier sur A et B pour sa taille et sa date de dernière modification. Si l'on a coïncidence dans les deux cas, le fichier n'a pas été modifié depuis la dernière synchronisation.
  - Dans le cas contraire on analyse le fichier par morceaux. Pour chaque morceau on calcule une *checksum*. Un noeud envoie ses *checksums* à l'autre qui les compare pour déterminer les blocks à envoyer.
  - Pour détecter des parties de fichier qui ont été simplement déplacés on calcule une deuxième checksum sur une fenêtre coulissante.
  - Et d'autres astuces encore ...

La sauvegarde de bases de données

Introduction

- À coté du stockage de blocs et stockage de fichiers, les bases de données sont une troisième manière d'organiser le stockage.
  - Possibilité de stocker des données avec une granularité fine et de les combiner dynamiquement en faisant des requêtes
  - Notion de transactions
  - Généralement les BD stockent leurs données dans des fichiers .



La sauvegarde de bases de données

Introduction

- Les bases de données sont plus difficiles à sauvegarder et restaurer que les systèmes de fichiers.
  - Elles sont plus complexes que les systèmes de fichiers.
  - Elles souvent interconnectées avec d'autres systèmes et les données qu'elles contiennent représentent souvent des transactions financières.
    - Il est souvent nécessaire de récupérer l'état des données à un point dans le temps très précis *(point-in-time recovery).*
  - Elles font souvent partie d'un système critique qui doit fonctionner 24h sur 24 → on ne peut pas les arrêter.

La sauvegarde de bases de données

Introduction

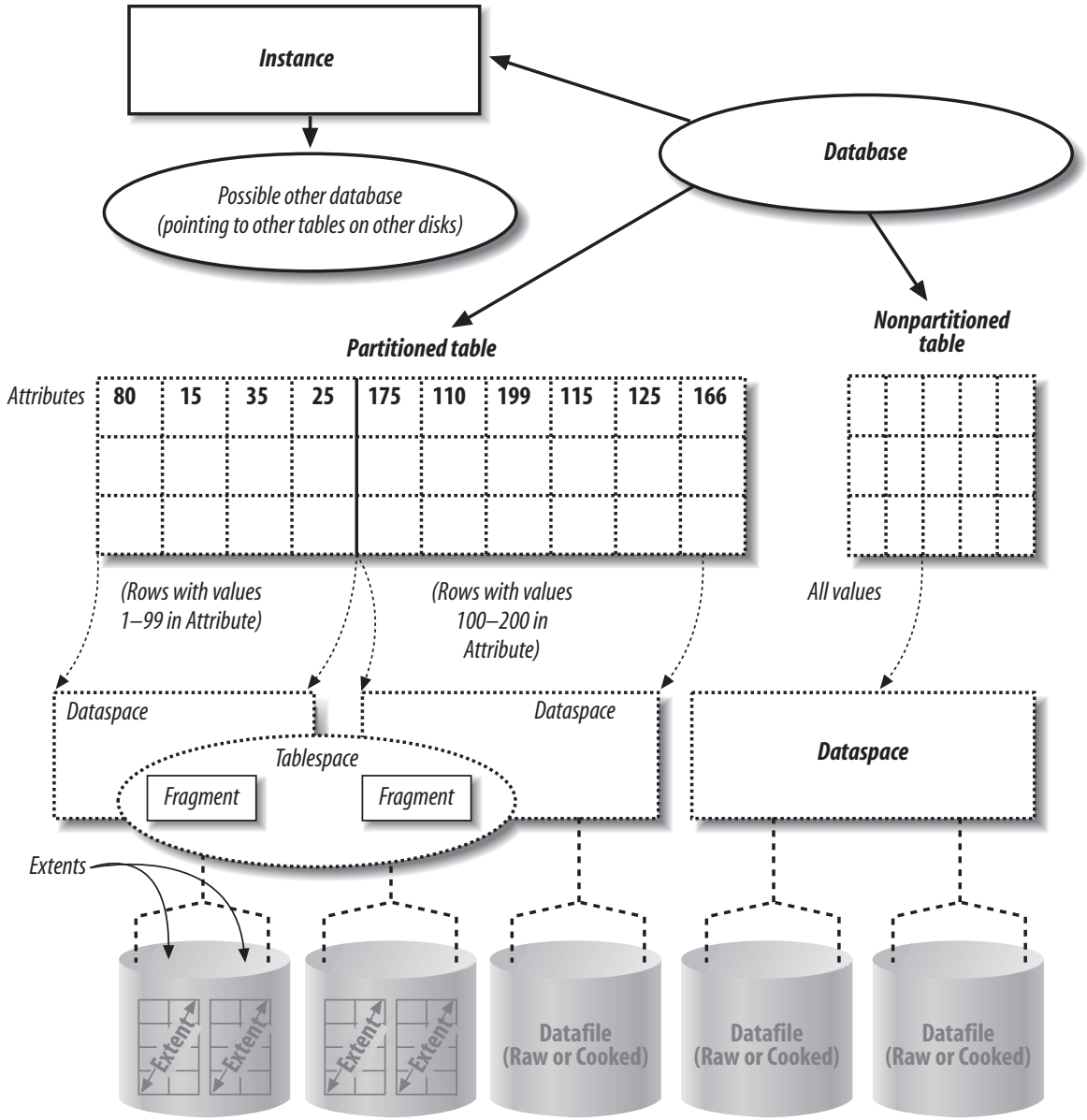
- Une BD est composée de plusieurs éléments qu'il faut tous sauvegarder :
  - Les données stockées dans la BD
  - Le système d'exploitation qui supporte la BD
  - Le logiciel de la BD
  - Le schéma de la BD

- Les principales options pour effectuer une sauvegarde des données d'une BD sont :
  - **Sauvegarde physique** : Sauvegarde des fichiers sous-jacents
    - en arrêtant la base de données : sauvegarde « froide »
    - en mettant la base de données dans un mode qui permet les sauvegardes : sauvegarde « chaude »
  - **Sauvegarde logique** : Sauvegarde des objets (tables) de la BD
    - Utilisation des outils de la base de donnés pour effectuer une sauvegarde dans un fichier
    - Utilisation d'un logiciel commercial type client-serveur qui offre un client pour la base de données

La sauvegarde de bases de données

Les éléments logiques d'une base de donné

- Notions importantes :
  - *Instance*
  - *Database*
  - *Table*
  - *Dataspace*
  - *Tablespace*
  - *Extent*
  - *Datafile*



La sauvegarde de bases de données

Les transactions

- Souvent les applications, surtout quand il y a de l'argent en jeu, demandent d'une base de données de fournir non seulement un stockage de données, mais aussi des transactions avec certaines garanties. Ce sont les propriétés ACID :
  - **Atomicité** *(atomicity)* : Une transaction se fait au complet (elle est *committed*) ou pas du tout (elle est annulée, *roll back*).
  - **Cohérence** *(consistency)* : Chaque transaction amène le système d'un état valide à un autre état valide.
  - **Isolation** *(isolation)* : L'exécution simultanée de transactions produit le même état que celui qui serait obtenu par l'exécution en série des transactions.
  - **Durabilité** *(durability)* : Lorsque une transaction a été confirmée *(committed)*, elle demeure enregistrée même à la suite d'une panne d'électricité, d'une panne de l'ordinateur ou d'un autre problème.

La sauvegarde de bases de données

Les transactions — Implémentation

- TID : Identificateur de transaction
- Inscription dans la BD seulement après le COMMIT
- Journalisation *(journaling)* :
  - Toute opération d'une transaction est notée avant et après l'exécution dans un fichier journal *(log)* présumé à l'abri de pannes
  - Les opérations de "commitment" sont notées avant d'être exécutées sur la BD *(write-ahead log protocol)*
- Points de reprise *(checkpoints)*
  - sauvegardes de l'état de la BD et notamment de TIDs de transactions en cours à intervalles réguliers

La sauvegarde de bases de données

Les transactions — Les *checkpoints*

- Écrire sur un disque est une opération beaucoup plus lente qu'écrire en mémoire.
  - Pour des accès aléatoires le disque est environ 100'000 fois plus lent que la mémoire.
- Pour améliorer la performance la BD essaie de garder les données le plus possible en mémoire et les écrit sur disque seulement quand c'est inévitable. (P. ex. pages modifiées, images des pages modifiées avant modification, transactions, ...)
- En cas de crash de la machine des données pas encore écrits sur disque seront perdues parce que la mémoire est volatile. Danger d'incohérence des fichiers.
- La BD a besoin d'un mécanisme pour retourner à un moment dans le temps où toutes les données étaient sur disque et rien était en mémoire. Ce moment s'appelle le ***checkpoint***.
  - À des intervalles réguliers la BD évacue toutes les données sur disque. Tous les *data files* et *log files* sont donc dans un état cohérent.
  - En cas de crash de la machine (sans dommage aux fichiers) on peut restaurer la BD au moment du *checkpoint*, puis répéter toutes les transactions commises depuis le *checkpoint* et annuler toutes les transactions incomplètes.
  - Chaque fois que l'administrateur informe la BD qu'il va commencer une sauvegarde des fichiers de la BD, celle-ci va faire un *checkpoint*.

La sauvegarde de bases de données

Les transactions — Les fichiers journal *(log files)*

- Un fichier journal *(log file)* est un fichier écrit séquentiellement par la BD sur disque pour survivre aux pannes
- Le journal peut contenir différents types d'articles avec leurs paramètres :
  - Début de transaction (TID)
  - Fin de transaction (TID, commit / abort)
  - Opération de transaction (TID, Opération, TupleId, BeforeImage, AfterImage)
  - Checkpoint record (Timestamp, Copie du data buffer au moment t, TID des transactions en cours au moment t)

La sauvegarde de bases de données

Les transactions — Déroulement d'une transaction

- Une base de données travaille généralement avec quatre espaces de stockage :
  - Le *datafile* : Fichier sur disque qui contient les tables
  - Le *rollback log* : Fichier journal sur disque qui permet d'annuler *(roll back)* une transaction
  - Le *transaction log* : Fichier journal sur disque qui permet de conclure une transaction interrompue
  - La mémoire partagée qui comporte
    - le *data buffer*
    - le *rollback log buffer*
    - le *transaction log buffer*

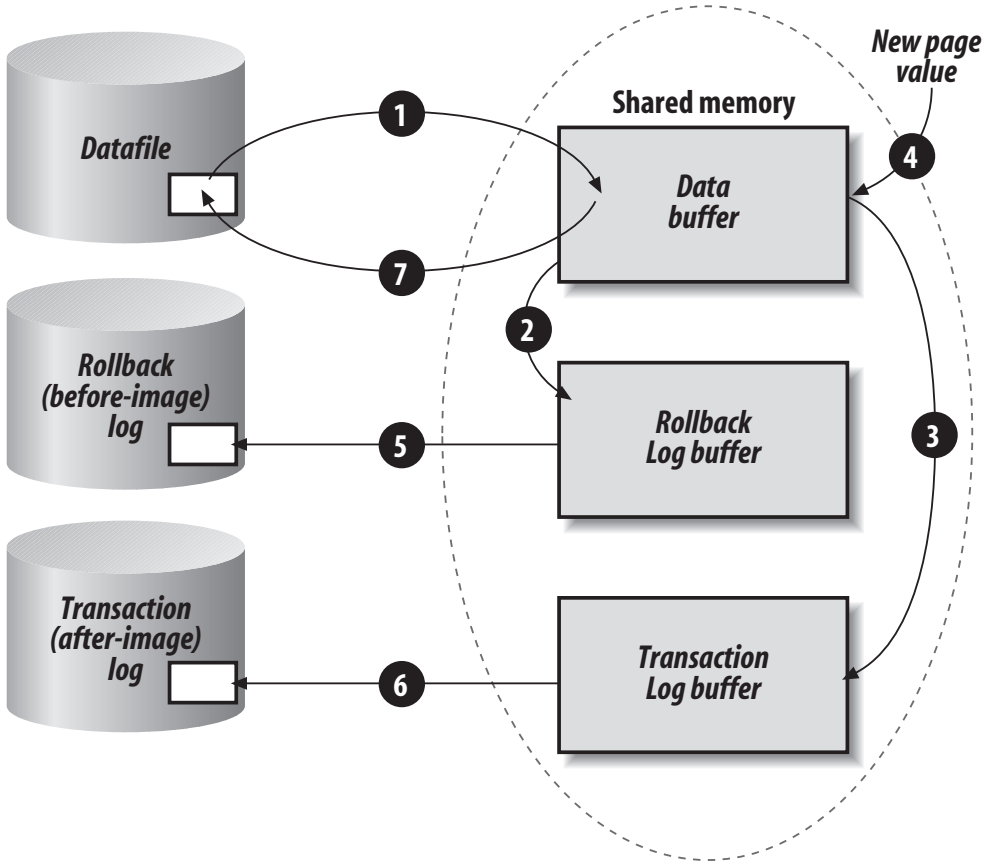
La sauvegarde de bases de données

Les transactions — Déroulement d'une transaction

- Avant toute chose le début de la transaction est noté dans le *transaction log*. Puis :

1. La BD lit la page qui va être modifiée et la met dans le *data buffer*.
2. Avant de commencer la transaction la BD sauvegarde une copie de la page **avant modification** dans le *rollback log buffer*.
3. La BD place une image de la page **après modification** dans le *transaction log buffer*.
4. La transaction peut maintenant modifier la page dans le *data buffer*.
5. Jusqu'ici, rien n'a été écrit sur disque. La BD commence les écritures disque en évacuant le *rollback log buffer* sur disque.
6. La BD évacue le *transaction log buffer* sur disque.
7. La BD évacue la page modifiée du *data buffer* sur disque.

- À un certain moment la transaction est *committed*. Ce fait sera immédiatement enregistré dans le *transaction log*.



La sauvegarde de bases de données

Les transactions — Algorithme général de reprise

- On trouve le dernier *checkpoint*
- On restaure le *data buffer* à partir de la copie dans le *checkpoint*
- On crée deux listes vides UNDO et REDO
- On lit le journal en arrière, et la liste des TIDs dans l'article *checkpoint* :
  - si Commit T, alors REDO := REDO + T ;
  - si Abort T, alors UNDO := UNDO + T ;
  - si Begin T et T $\notin$ REDO, alors UNDO := UNDO + T ;
- On défait les transactions dans UNDO
- On refait les transactions dans REDO