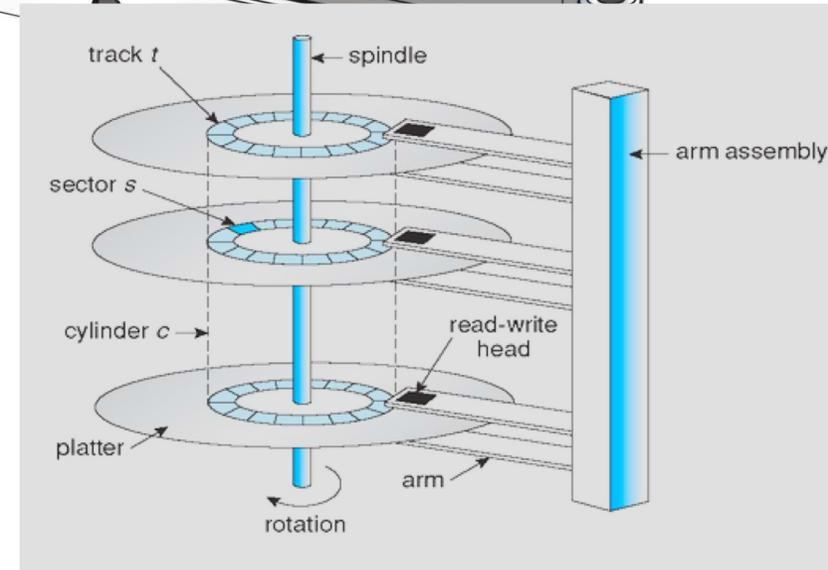
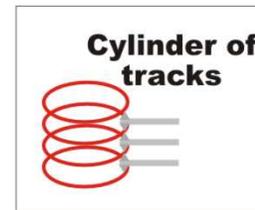
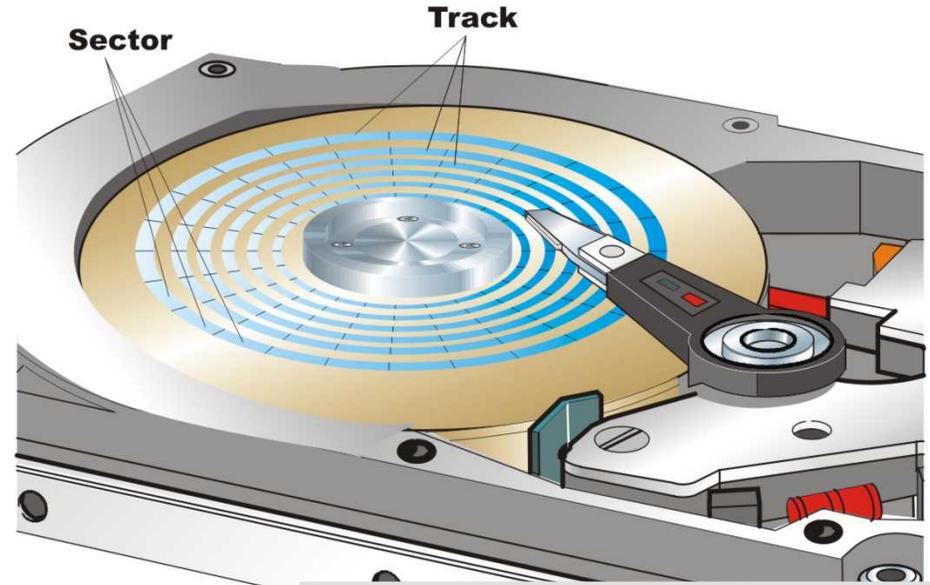


Mémoire en Masse

Démonstration

Disque Dur (HDD)

- Les disques magnétiques fournissent l'essentiel du stockage secondaire des ordinateurs modernes
 - **Transfer rate (taux de transfert) (R_s):** taux auquel les données sont transférées entre le lecteur et l'ordinateur
 - **Seek time (temps de chercher) (T_a):** temps de déplacement de la tête.
 - **rotational latency (latence de rotation) (T_l):** temps de déplacement du bloc par rotation
 - Têtes volent sur les plateaux
 - **Head crash (accident de la tête):** résultat de la tête en contact avec la surface du disque (pas bon!)
- Track (Piste) = platter (plateau) \times cylinder (cylindre)



[Super vidéo de comment les SSD/la mémoire flash fonctionne](#)

Disque Dur (HDD)

- **IOPS (I/O Operations per second):** La mesure est généralement utilisée pour décrire la capacité d'un périphérique de stockage à traiter des demandes d'entrée-sortie aléatoires.
 - Une "opération" est équivalente au transfert d'un "bloc" entier
 - **Block size (taille d'un bloc):** quantité de données transférées pendant une opération d'E/S (N secteurs: e.g., secteur = 512B, bloc = 4KB)
- **Throughput (débit):** nombre de bytes transférés par unité de temps
- **Response time (temps de réponse) (T_r):** temps nécessaire pour traiter une demande d'opération d'E/S
- **Transfer time (T_t):** temps pour transférer **un bloc**

$$\text{IOPS} = \text{throughput} / \text{block_size}$$

$$T_{i0} = T_a + T_l + T_t = 1/\text{IOPS}$$

$$T_t = \text{block_size} / R_s$$

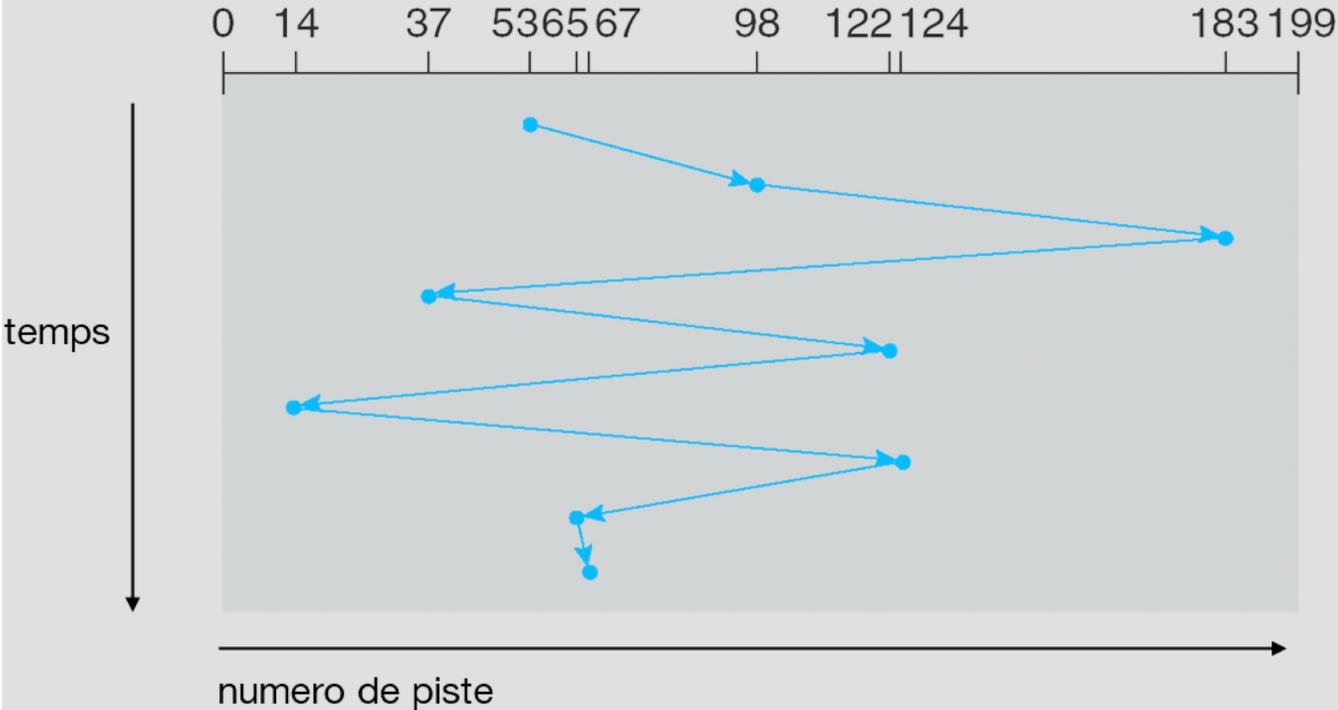
$$\text{throughput} = R_s * T_t / T_{i0}$$

$$\text{cas optimal: } T_r = T_{i0}$$

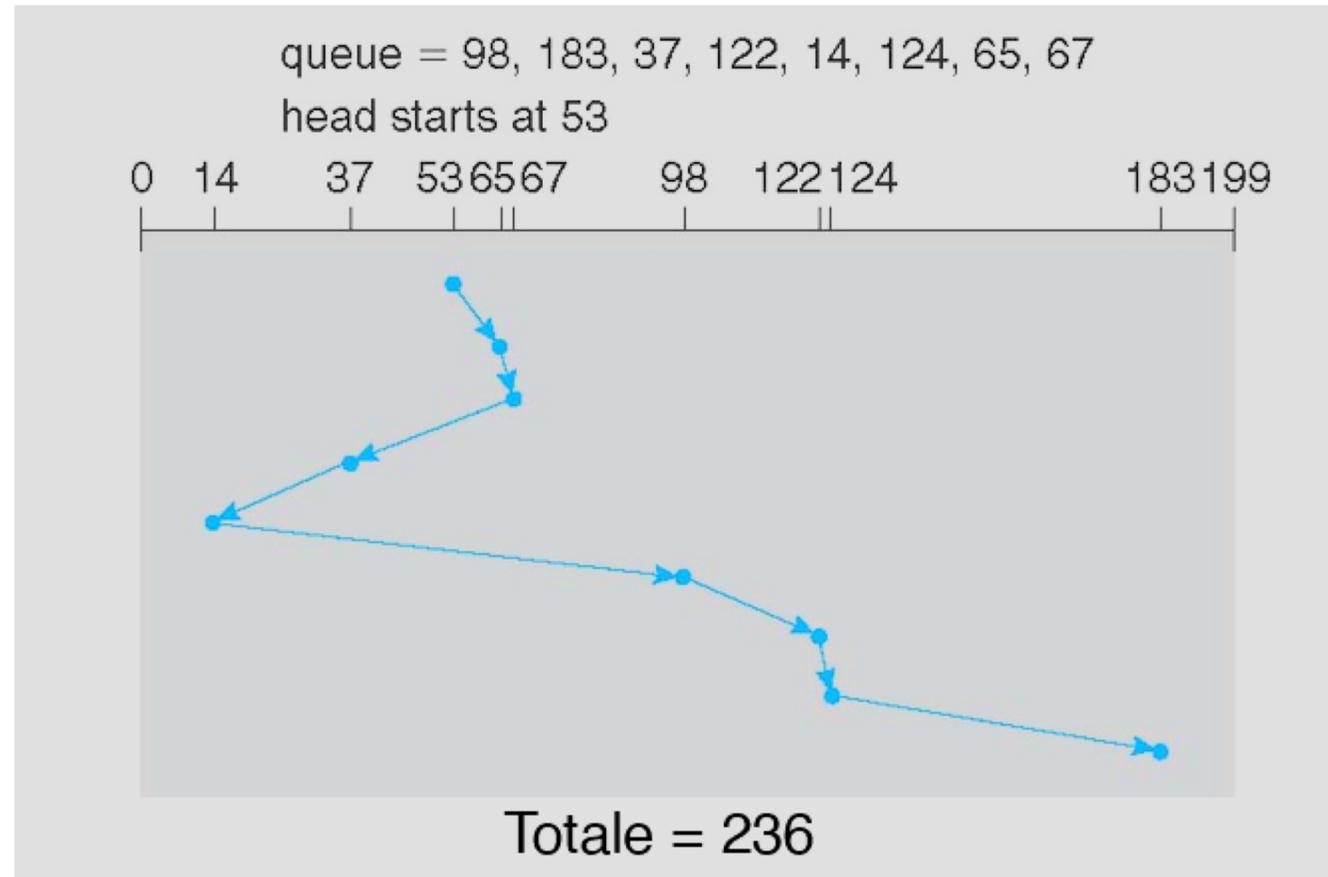
Greedy (on l'adore)! FCFS

First-come first-served (FCFS)

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53

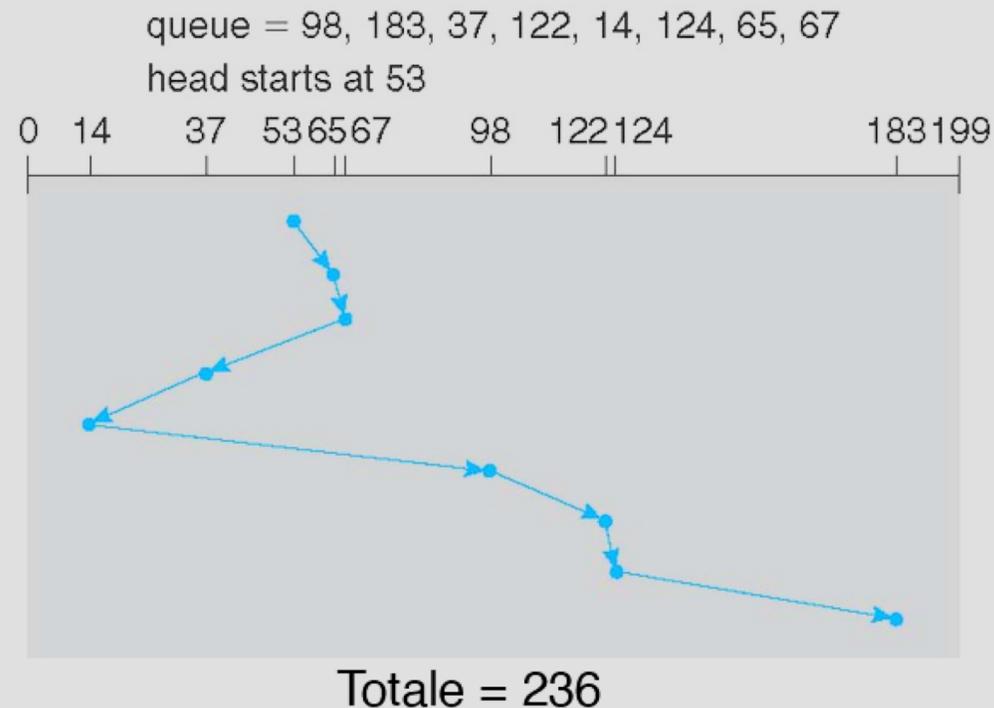


SSTF



SSTF

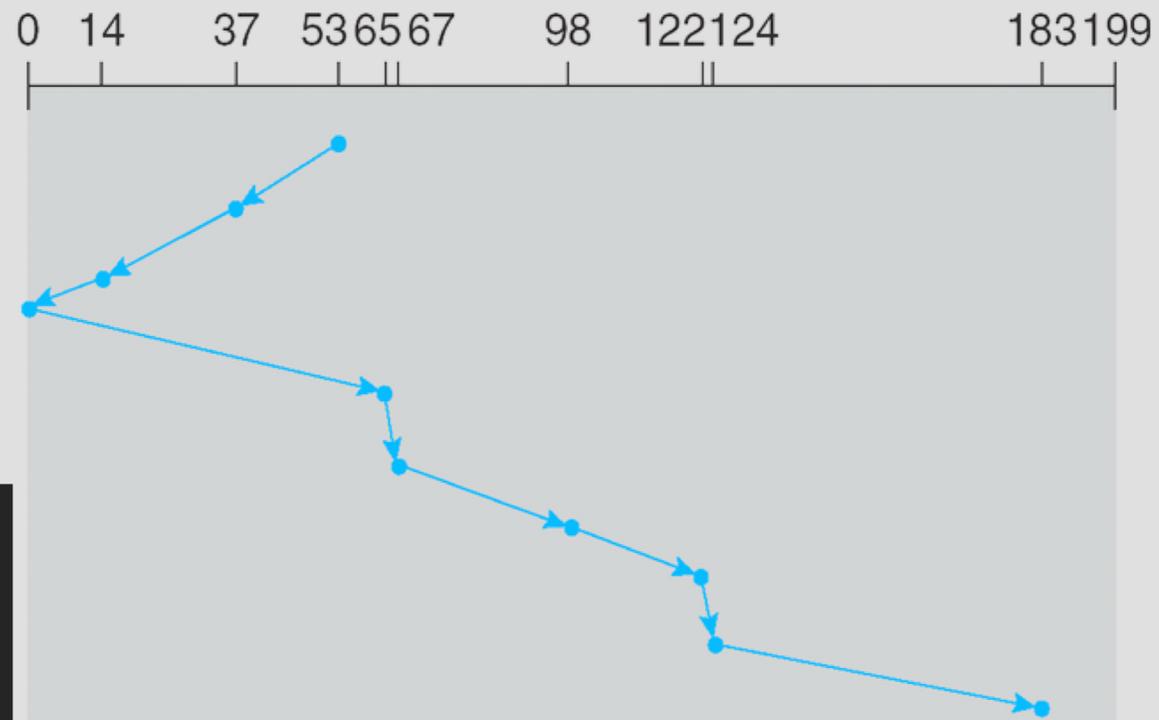
- **Shortest Seek Time First** sélectionne la requête avec le “seek time” minimum à partir de la position de tête actuelle
- Problème? Famine.
- Meilleur que FCFS, mais pas optimal



SCAN

- La tête parcourt toute la surface dans un sens puis dans l'autre

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53

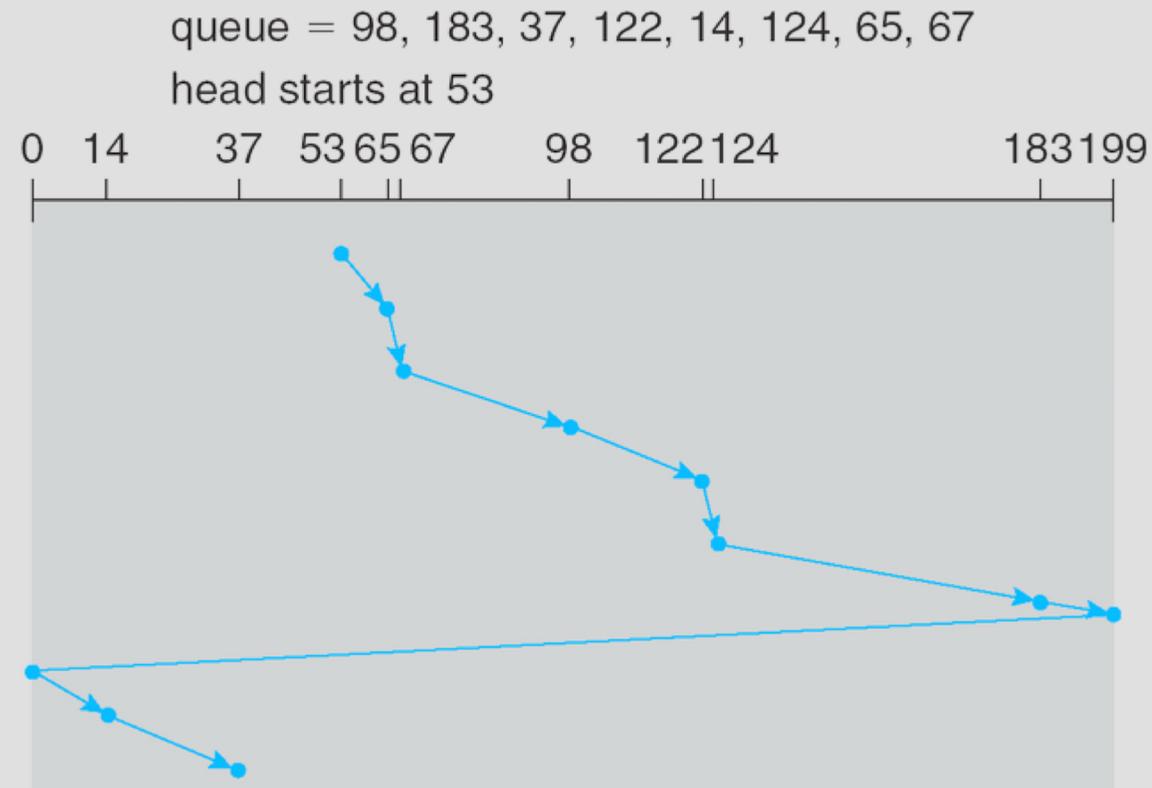


Totale = 236



C-SCAN

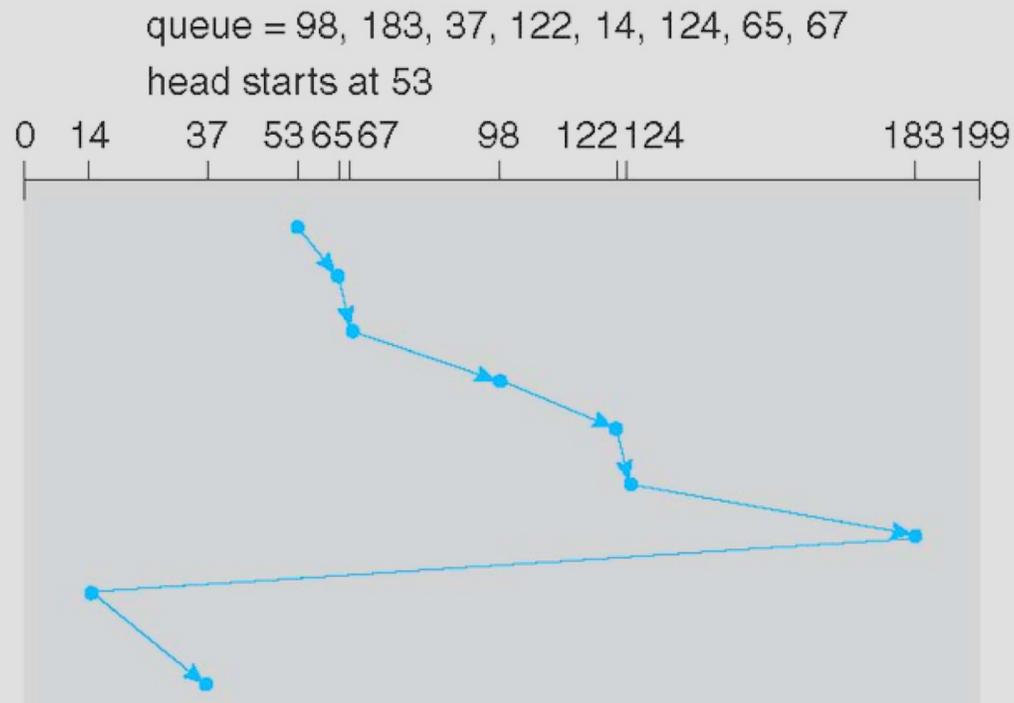
- La tête parcourt la surface toujours dans le même sens
- Diminue l'attente maximum par rapport à SCAN



Déplacement total plus grand (382) mais $1 \times 200 < 2 \times 100$

LOOK et C-LOOK

- LOOK = SCAN sans aller vraiment jusqu'au bout
- C-LOOK = C-SCAN en évitant aussi les déplacements jusqu'au bout



Exos!

Question 3. Mémoire en masse (8 points au total)

(a) (4 points) Un disque a 120 pistes. À l'instant présent, la tête de lecture est sur la piste 38 en se déplaçant dans le sens de l'augmentation des numéros de piste. La séquence des requêtes de lecture de piste à venir est:

[80, 20, 10, 110]

Quel est la **distance totale parcourue** par la tête de lecture pour les algorithmes d'ordonnancement de disques suivants:

- i. FIFO
- ii. Shortest Seek Time First (SSTF)
- iii. C-LOOK

(b) (2 points) Compte tenu des informations suivantes:

- Taille d'un bloc est **32 KB**
- Le disque tourne a une vitesse de **7200 révolutions par minute**
- Le taux de transfert est **5 Mb/s** (N.B. $b \neq B$)
- Le seek time moyen est **9 ms**

- i. (0.5 points) Quel est le temps de transfert pour un bloc (en ms)? **50**
- ii. (0.5 points) Quel est la latence rotationelle moyen (en ms)? **4.17**
- iii. (1 point) Quel est le temps **total moyen** pour transférer un bloc (en ms)? **63.17**

Bits de parité

- 1B -> a un 1b de parité
- Nombre de bits pair dans le byte -> bit à 0
- Nombre de bits impair dans le byte -> bit à 1
- On a donc besoin de 1b de plus pour chaque 1B dans la mémoire

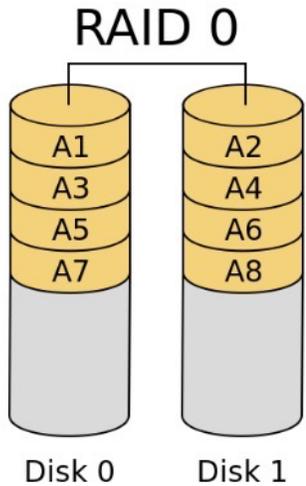
(c) (2 points) Avec un système avec codage de Hamming à parité paire calculée de gauche à droite, vous recevez le code de Hamming 12 bits suivant:

0 0 1 1 1 0 0 1 0 1 1 1

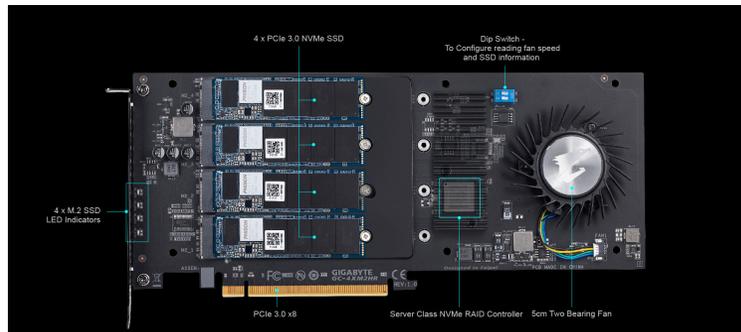
Ce code a une erreur de 1 bit, quel bit ?

RAIDs

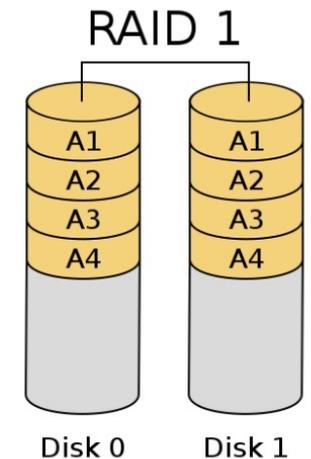
Permettent d'améliorer la fiabilité, la vitesse, ou la "grandeur" de disques (ou de SSD).



RAID-0: Permet de traiter n disques distincts comme un seul contigu. Ça tue la fiabilité ($1/\text{fiabilité}(\text{disque}_i)$)! Mais la vitesse de lecture/écriture est très grande (somme de celle des disques simples)



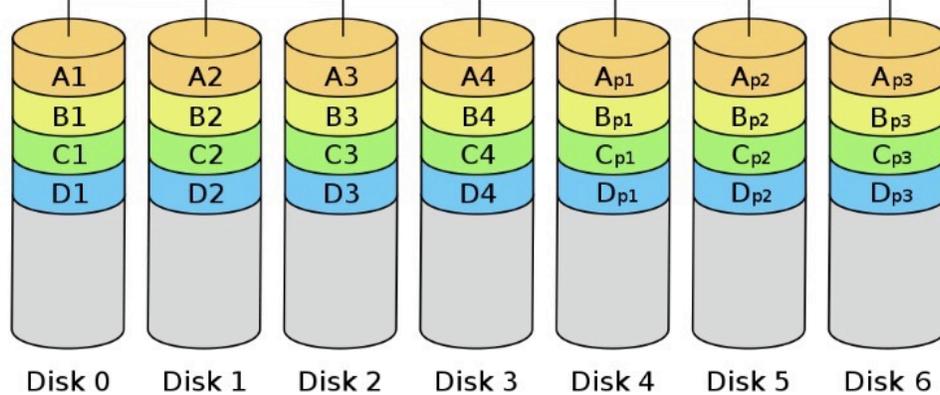
RAID-1: Clone le disque n fois. C'est donc $n * \text{fiabilité}(\text{disque}_i)$ de fiabilité! La lecture est TRÈS bonne (on peut facilement dédier certains disques à certains blocs, c'est comme si on a n têtes de lectures en un sens), et l'écriture est équivalente à avoir aucun RAID.



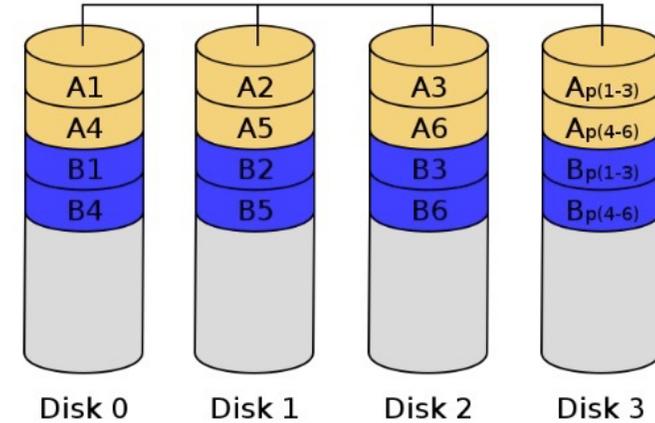
RAIDs

Permettent d'améliorer la fiabilité, la vitesse, ou la "grandeur" de disques (ou de SSD).

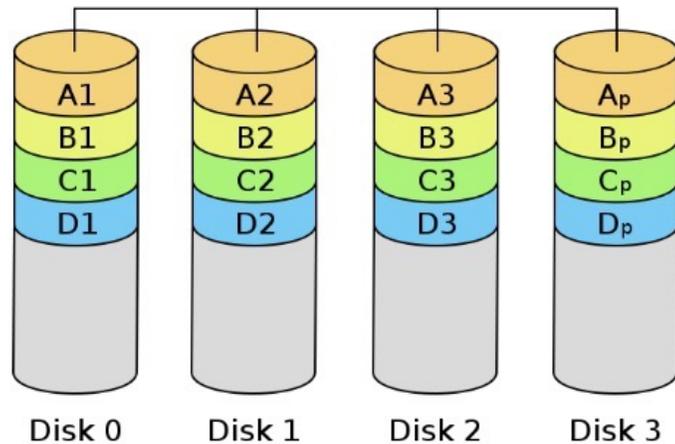
RAID 2



RAID 3



RAID 4



RAID-2-4: on fait RAID-0 sur n-1 disques, et on utilise le nième pour un calcul de parité.

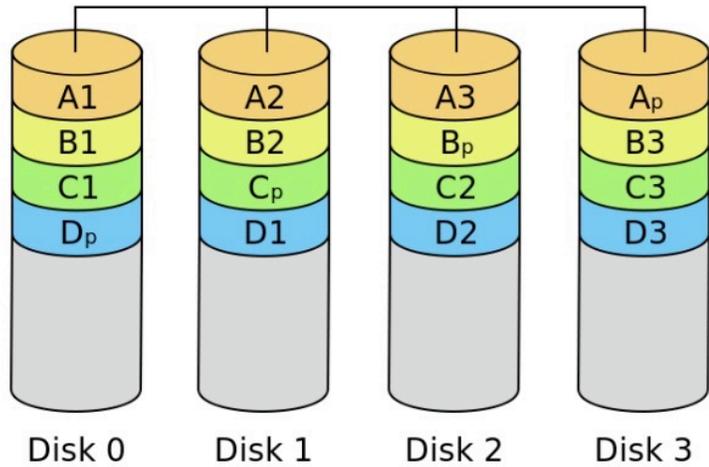
- Lecture très bonne, mais écriture très mauvaise: on doit vérifier le disque de parité à chaque écriture, ce qui implique souvent de faire d'autres lectures ou écritures sur le nième disque.

- Parfois, si on a un error-correcting code, ça implique d'aller corriger les autres disques, ce qui implique aussi d'autres écritures...

RAIDs

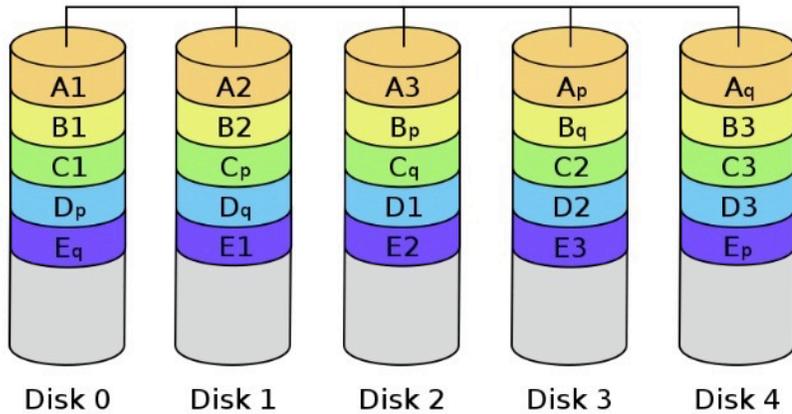
Permettent d'améliorer la fiabilité, la vitesse, ou la "grandeur" de disques (ou de SSD).

RAID 5



RAID-5: comme RAID-2-4, mais tous les disques sont utilisés pour le stockage, et un leur réserve une section pour la parité.

RAID 6

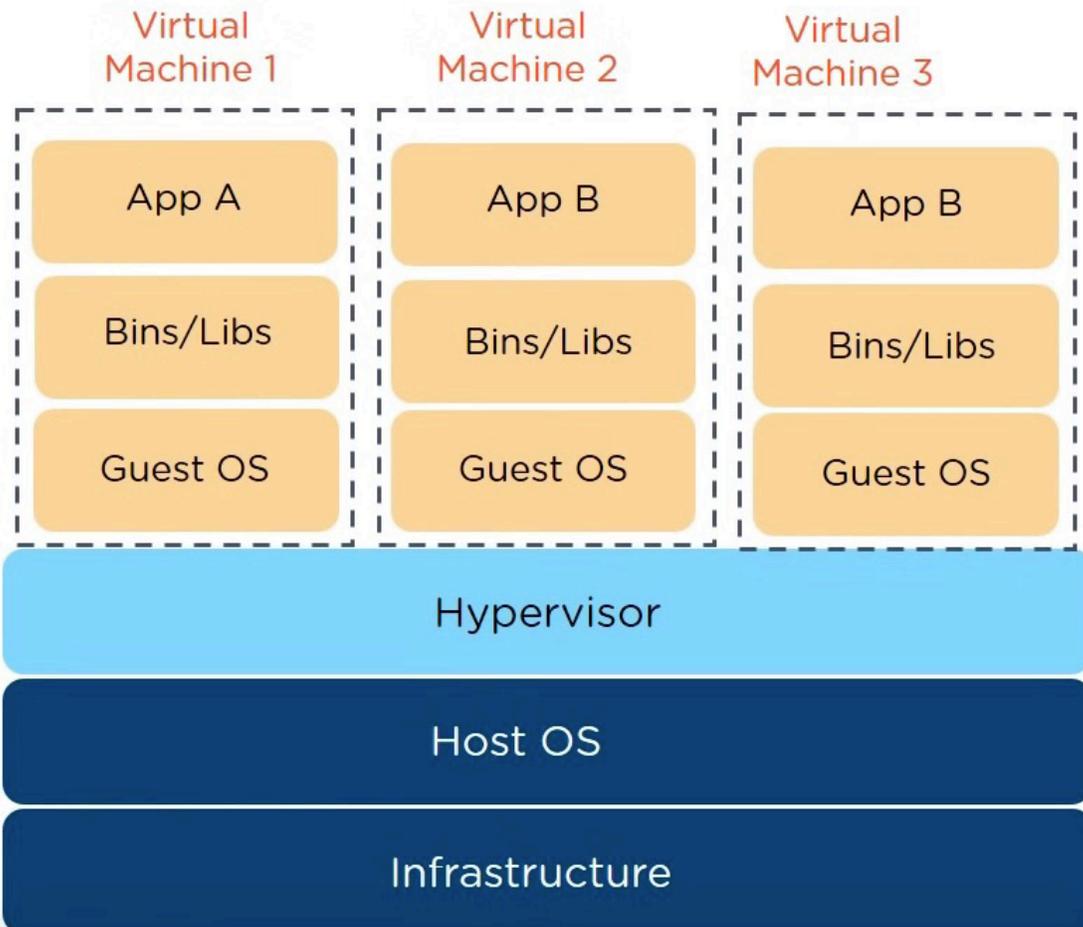


RAID-6: on ajoute des bits de correction pour pouvoir récupérer un disque qui flanche.

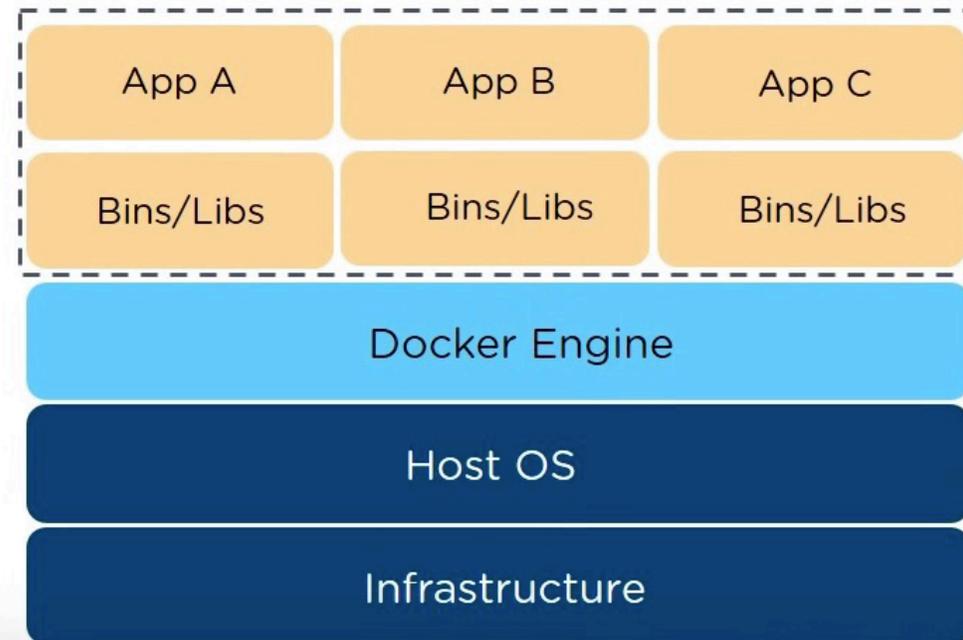
DOCKER!!!!!!!



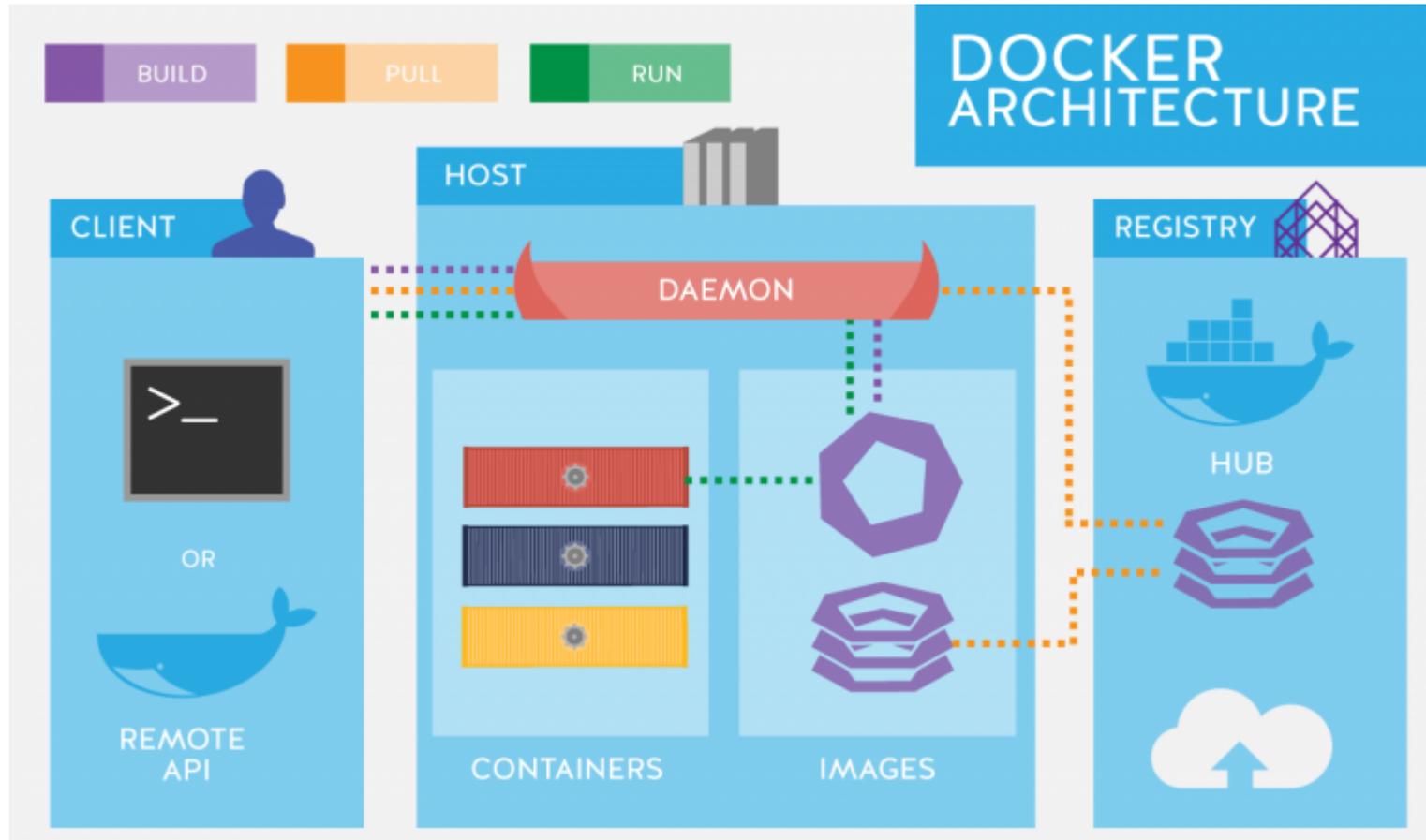
Virtual Machines



Docker Container



DOCKER!!!!!!!



[Docker in 100s](#)

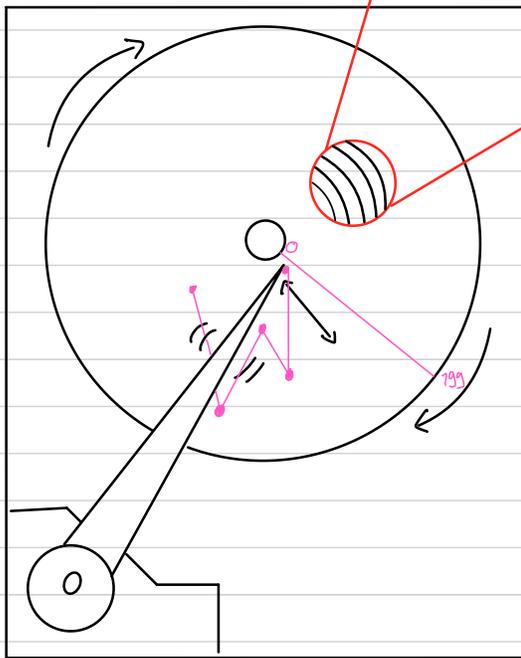
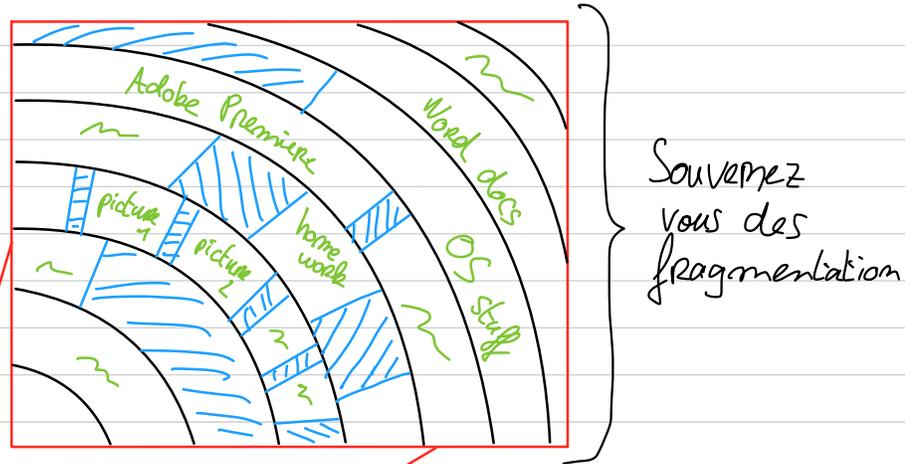
[Learn Docker in 10mins](#)

DOCKER!!!!!!

Dockerfile de votre autograder et image pour ceux qui utilisent macOS pour les devoirs

Mémoire en masse

- Bien visualiser



$$\begin{aligned} \text{IOPS} &= \text{throughput} / \text{block_size} \\ T_{io} &= T_a + T_l + T_t = 1/\text{IOPS} \\ T_t &= \text{block_size} / R_s \\ \text{throughput} &= R_s * T_t / T_{io} \\ \text{cas optimal: } T_r &= T_{io} \end{aligned}$$

- Performances disques → formules sur fiche!
↳ que veulent-dire les variables!

- Connaissez les algos
 - FCFS
 - Shortest - seek - time - first (famine)
 - Scan (scanning mouvement: donc touche bord)
 - Look (cherche spécifiquement)
 - C-Scan } même mais 1 seul sense! Lata retourne
 - C-Look }

• Connaissez + Comprenez vos RAIDs !!

• Bits de parité :

that might be starting to fail or has already failed.

Memory systems have long detected certain errors by using parity bits. In this scenario, each byte in a memory system has a parity bit associated with it that records whether the number of bits in the byte set to 1 is even (parity = 0) or odd (parity = 1). If one of the bits in the byte is damaged (either a 1 becomes a 0, or a 0 becomes a 1), the parity of the byte changes and thus does not match the stored parity. Similarly, if the stored parity bit is damaged, it does not match the computed parity. Thus, all single-bit errors are detected by the memory system. A double-bit-error might go undetected, however. Note that parity is easily calculated by performing an XOR (for "eXclusive OR") of the bits. Also note that for every byte of memory, we now need an extra bit of memory to store the parity.

Parity is one form of checksums, which use modular arithmetic to

• Hamming code

Entraînement !

- Bon les exos sur les algos ^{des disques} V sont pretty straight forward mais vous pouvez demander si vous avez un doute (no shame!)

(c) (2 points) Avec un système avec codage de Hamming à parité paire calculée de gauche à droite, vous recevez le code de Hamming 12 bits suivant:

0 0 1 1 1 0 0 1 0 1 1 1

Ce code a une erreur de 1 bit, quel bit ? \bar{P}_2 \bar{P}_4 \bar{P}_8

On nous a envoyé un code de Hamming, donc les bits de parités sont déjà dans la chaîne :

P_1 : 001110010111

P_1 est ici

P_1 c'est, on prend 1, skip 1, repeat

On a donc $P_1 = 011001$

= 1 car $n \cdot 1$ est impair

Rappel (voir plus haut) : on compte les 1

impair $\rightarrow 1$
pair $\rightarrow 0$

P_2 : 001110010111

P_2 est ici

Prend 2, skip 2, repeat

On a donc $P_2 = 010011 = 1$

Tip Δ : Ici, ou bien dans le Hamming coding,

si vous prenez un autre bit de parité

dans vos "prendre x , skip x , repeat",

alors vous avez mess up vos "take + skip".

Donc marquer tout de suite $P_1, P_2, P_4 \dots$ sur la chaîne.

P_4 : Prend 4, skip 4, repeat (you get the deal)

0 0 1 1 1 0 0 1 0 1 1 1

P_4

$$P_4: 11001 = 1$$

P_8 : Prend 8, skip 8

0 0 1 1 1 0 0 1 0 1 1 1

P_8

$$P_8: 10111 = 0$$

Finalement

$$P_1: 011001 = 1 \quad 1$$

$$P_2: 010011 = 1 \quad + 2$$

$$P_4: 11001 = 1 \quad + 4$$

$$P_8: 10111 = 0 \quad + 0 \text{ car pas } 1$$

" 7

Si on veut fix la chaîne? On switch juste le bit 7 😊

0 0 1 1 1 0 0 1 0 1 1 1 1

0 0 1 1 1 0 1 1 0 1 1 1

et pour l'octet correct on retine les bits de parité souligné

Fichiers

Question 4. Système de fichiers (4 points au total)

(a) (3 points) Un système de fichiers ext2 utilise un adressage 32-bit et une taille de bloc de 2 KB.

i. (1 point) Quelle est la taille maximale qu'un fichier peut avoir pour qu'il n'ait pas besoin d'utiliser le pointeur simple (*single*) indirection dans l'inode (en B, peut être une expression) ?

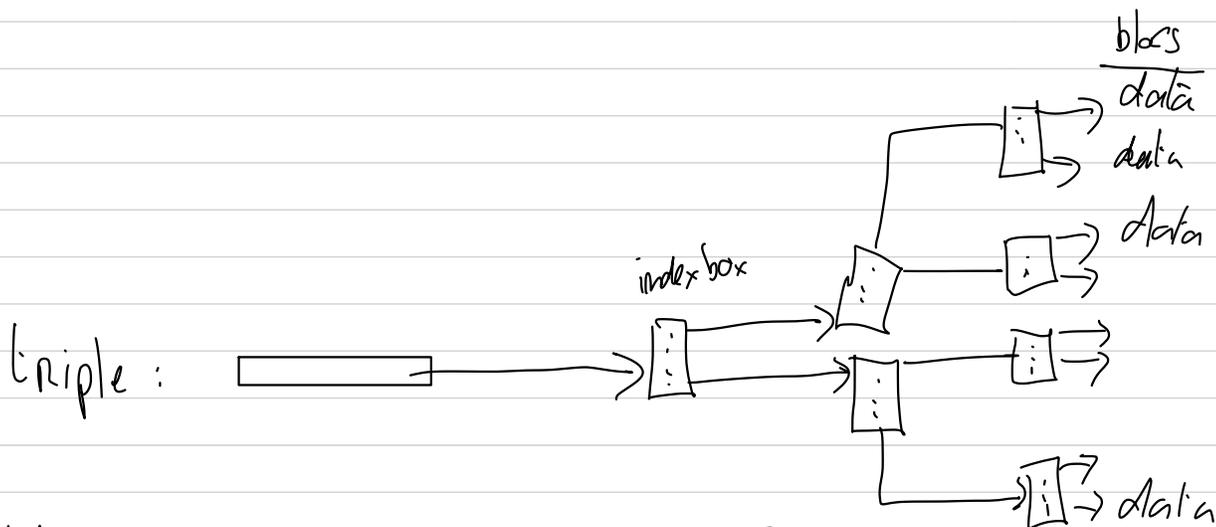
ii. (1 point) Quelle est la taille maximale qu'un fichier peut avoir pour qu'il n'ait pas besoin d'utiliser le pointeur double indirection dans l'inode (en B peut être une expression) ?

iii. (1 point) Quelle est la taille maximale qu'un fichier peut avoir pour qu'il n'ait pas besoin d'utiliser le pointeur triple indirection dans l'inode (en B, peut être une expression) ?

(b) (1 point) Cela décrit le mieux quel type de système de fichiers ? " Similaire à un système de fichiers qui est en chaîne mais avec un meilleur accès aléatoire "

Rappel ext utilise 12 pointeurs pour direct
3 autres pointeurs pour single, double et triple

$$2 \text{ KB} = ? \quad 2 = 2^1 \quad \text{KB} = 2^{10} \text{ donc } 2^{11} \text{ B}$$



$$\text{Adresse de 32 bits} = 4 \text{ B} = 2^2 \text{ B}$$

donc nombre de rangées possibles
dans une index box avec des
adresses de 32b ? $\Rightarrow \frac{2^{11}}{2^2} = 2^9$

(b) (2 points) Compte tenu des informations suivantes:

- Taille d'un bloc est **32 KB** *block-size*
- Le disque tourne a une vitesse de **7200 révolutions par minute**
- Le taux de transfert est **5 Mb/s** (N.B. $b \neq B$) *R_s*
- Le seek time moyen est **9 ms** *T_a*

- (0.5 points) Quel est le temps de transfert pour un bloc (en ms)?
- (0.5 points) Quel est la latence rotationelle moyen (en ms)?
- (1 point) Quel est le temps **total moyen** pour transférer un bloc (en ms)?

Question exan (b)

Convert bloc size to bits

$$1 B = 8 b$$

$$1 KB = 2^{10} B = 1024 B$$

$$\text{Donc } 32 KB = (32 \times 1024) \times 8 = 262144 b$$

$$\text{Taux de transfert: } 5 Mb/s = 5 \times 2^{20} b$$

⚠ $b \neq B$

$$T_t = \frac{262144}{5 \times 2^{20}} = 0.05 s = 50 ms$$

$$\text{ii) } 1 \text{ min} = 60 s \text{ donc } \frac{60}{7200} \text{ minutes per rotation}$$

$$\underline{\text{Latence rotationelle moyenne}} = \frac{60}{7200} \times \frac{1}{2}$$

↳ en mins

60 000 ms dans une minute

Donc

$$\begin{aligned} \text{Latence rotationnel moyenne} &= \frac{60}{7200} \times \frac{1}{2} \times 60000 \\ &= 4.17 \text{ ms} \end{aligned}$$

iii)

$$\begin{aligned} \text{temps total moyen} &= \text{Avg seek time} \\ &\quad + \text{Avg rot latency} \\ &\quad + \text{Transfer time} \end{aligned}$$

$$50 + 4.17 + 9 = 63.17$$