

## Séance 6

# Gestion du stockage et mémoire cache



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

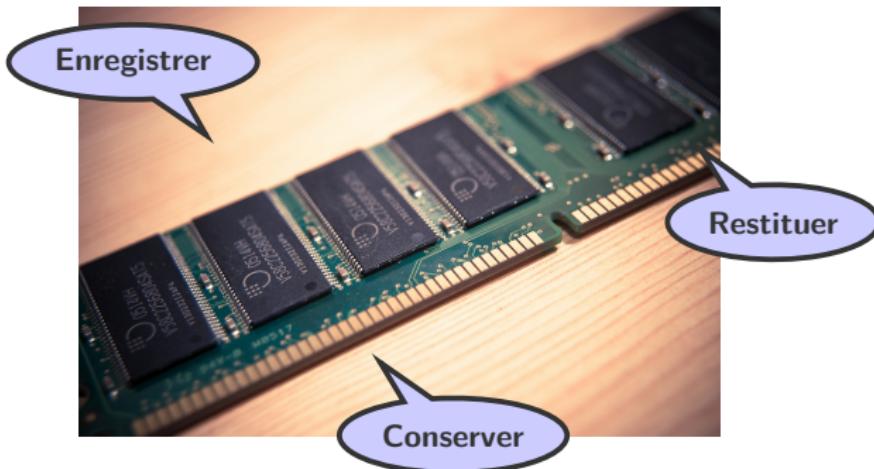
- Types de mémoire et **stockage hiérarchisé**
  - Hiérarchie, types et caractéristiques des mémoires
  - Technologie RAID
- Fonctionnement et utilisation de **mémoires caches**
  - Analyse de la performance des caches
  - Principes de fonctionnement des mémoires caches
  - Problèmes de consistences de données

# Stockage hiérarchisé



# Mémoire

- Deux **caractéristiques essentielles** pour un ordinateur
  - **Vitesse de traitement** d'un grand nombre d'information
  - **Capacité de stockage** des informations



# Hiérarchie des mémoires (1)

- Temps d'accès à la mémoire, en lecture et écriture

*Peu de mémoire très rapide, car très couteuse*

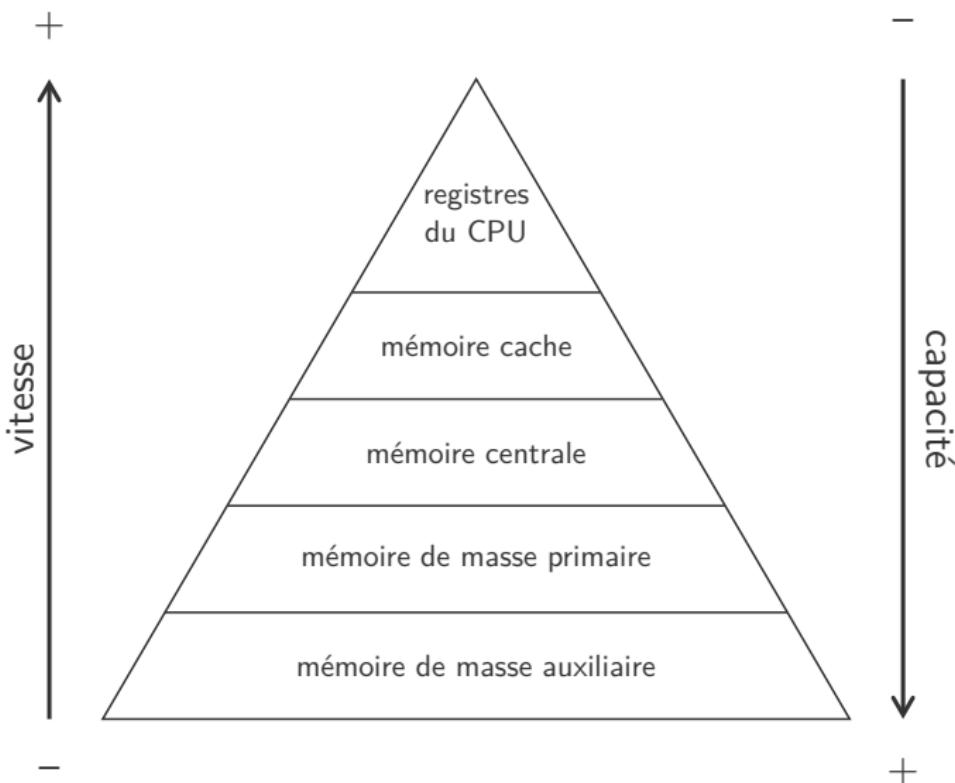
- Capacité de stockage de la mémoire

*Mémoire peu rapide disponible en grande quantité*

- Déplacement du CPU vers les mémoires de masse

*Temps d'accès et capacité augmentent, **cout par bit** diminue*

## Hiérarchie des mémoires (2)



# Types de mémoires (1)

- **Registres** du CPU ont une grande vitesse

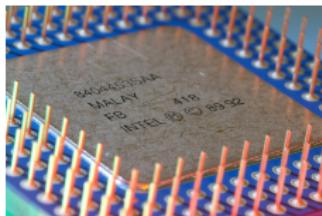
*Stockage des opérandes et résultats intermédiaires*

- **Mémoire cache** rapide et de faible capacité

*Tampon entre le CPU et la mémoire centrale*

- **Mémoire centrale** plus grand temps d'accès

*Principale zone de stockage utilisée par le CPU*



# Types de mémoires (2)

- Mémoire de **masse principale**

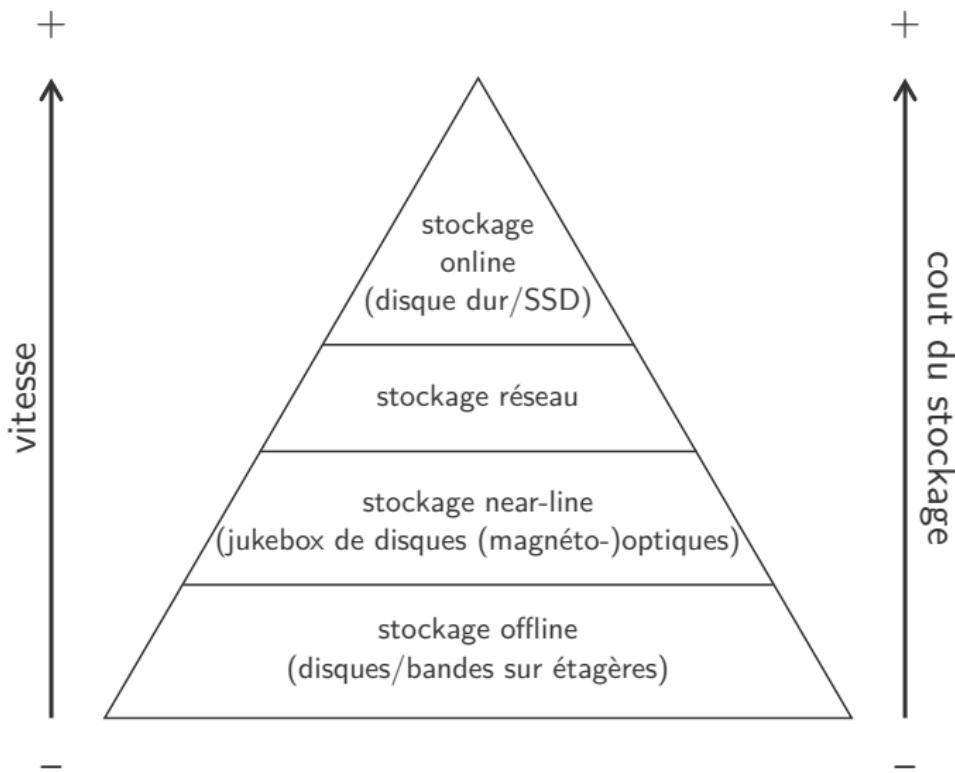
*Mémoire de stockage permanente d'informations*

- Mémoire de **masse auxiliaire**

*Stockage permanent de backup et archivage, à long terme*



# Mémoire de masse



# Types de mémoires de masse

- **Quatre niveaux** principaux de stockage de données
  - **Online** : accès quasi instantané à l'information
  - **Réseau** : accessible à travers un réseau local ou distant
  - **Near-line** : accès par un robot aux disques
  - **Offline** : stocké sur des étagères et accès humain



# Caractéristique des mémoires

Média	Temps d'accès	Débit	Capacité
Registres	1 ns		4 ko
Mémoire cache	2 ns		64 Mo
Mémoire RAM	5–60 ns	1–20 Go/s	512 Mo–16 Go
Disques durs	3–20 ms	10–320 Mo/s	250 Go–4 To
Disques magnéto-optiques	15–40 ms	10 Mo/s	2–9,1 Go
Clés USB	10 ms	25 Mo/s	1–64 Go
CD	120 ms	1–8 Mo/s	650 Mo
DVD	140 ms	2–22 Mo/s	4,7–17 Go
Jukebox médias	Quelques secondes	Vitesse des lecteurs	To–Po
Cassette 3592	Quelques secondes	5 Mo/s	4 To
Disques SSD	0,1 ms	500 Mo/s	128–512 Go

# Le rêve... et l'illusion

- Besoin d'une **quantité illimitée de mémoire rapide**

*Le rêve de tout programmeur qui n'aura ainsi plus de limites*

- Combinaisons de mémoire pour donner l'**illusion du rêve**

*Pas besoin de tout, tout de suite et avec la même probabilité*

- Deux différents types de **localités**

- **Temporelle** : un élément référencé le sera bientôt de nouveau
- **Spatiale** : les éléments référencés sont proches en adresses

# Hiérarchie de mémoire

- Établissement d'une **hiérarchie de mémoires** dans un système

*Combinaison de technologies avec différentes caractéristiques*

- **Transfert de données** entre deux couches adjacentes

*Attention à prendre en compte le cout du transfert*

Vitesse	Taille	Cout (\$/bit)	Technologie actuelle
Plus rapide	Mémoire	Plus petite	Plus grand
	Mémoire		SRAM
Plus lent	Mémoire	Plus grande	DRAM
	Mémoire	Plus bas	Disque magnétique

# Technologie RAID (1)

- Redundant Arrays of Independent Disks (RAID)

*Batteries de disques durs pour meilleure capacité et sécurité  
Gestion logicielle ou matérielle*

- Plusieurs **schémas** possibles de RAID 0 à RAID 6

- **RAID 0** : aucune redondance
- **RAID 1** : données dupliquées (mirroring)
- **RAID 2** : code de Hamming
- **RAID 3 à RAID 5** : simple calcul de parité

- **Récupération** des données lors du crash d'un disque

*Et de plusieurs disques pour RAID 6*

# Technologie RAID (2)

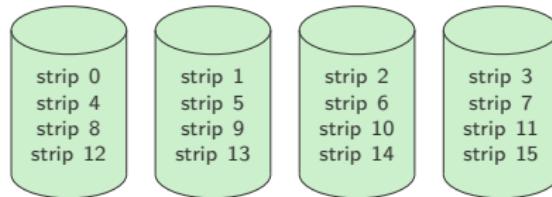
- **Bénéfices** de la technologie RAID
  - Amélioration de la fiabilité par redondance
  - Amélioration de la performance via parallélisme
- **Mirroring** consiste à simplement recopier les données

*Permet de doubler la vitesse de lecture*
- **Striping** consiste à séparer les bits d'un octets
  - Chaque bit d'un octet est placé sur un disque différent
  - Temps d'accès identique à un seul disque
  - Chaque accès rapatrie huit fois plus de données

# RAID 0

- Non-redundant striping
  - Nécessite  $N$  disques
- Striping au niveau des blocs mais sans redondance

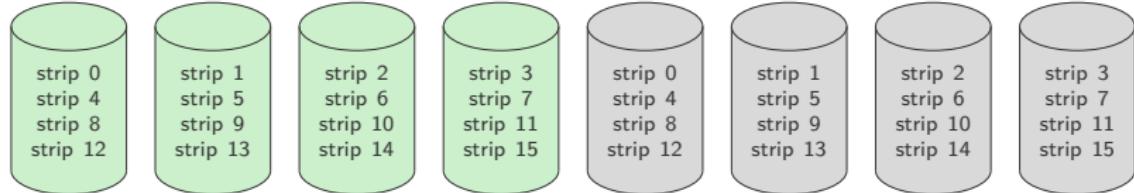
*Blocs d'un même fichier répartis sur plusieurs disques*



# RAID 1

- Mirrored disks
  - Nécessite  $2N$  disques
- Mirroring des disques, **copies complètes** en double

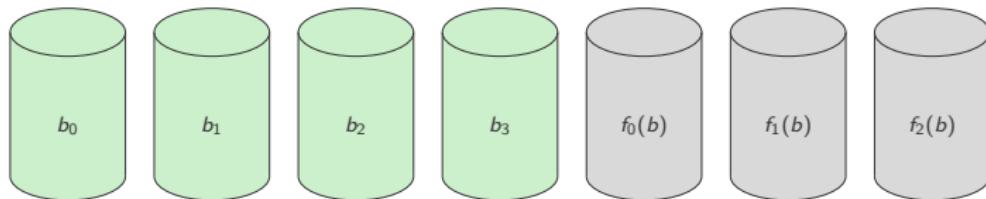
*Peut être combiné avec du striping*



# RAID 2

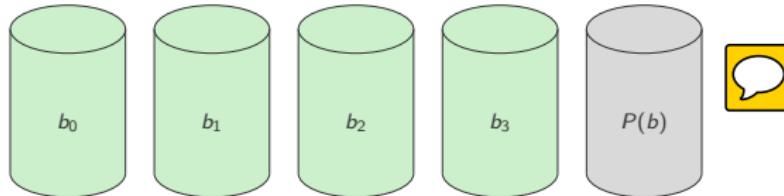
- Accès parallèle avec **memory-style error-correcting codes**
  - Nécessite  $N + m$  disques
- Stockage d'informations sur la **parité** (ECC)

*Code de Hamming pour pouvoir réparer un bit endommagé*



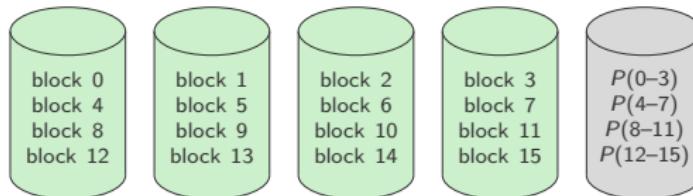
# RAID 3

- Accès parallèle avec **bit-interleaved parity**
  - Nécessite  $N + 1$  disques
- **Un seul bit** de parité est suffisant
  - Car contrôleur disque vérifie déjà les secteurs avec un ECC
  - Calcul et écriture de la parité du CPU vers contrôleur RAID



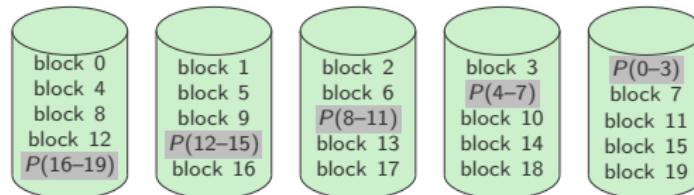
# RAID 4

- Accès indépendant avec **block-interleaved parity**
  - Nécessite  $N + 1$  disques
- **Striping au niveau des blocs** avec un disque de parités
  - Accès à un seul disque lorsque lecture d'un seul bloc
  - Lecture de plusieurs blocs en parallèle



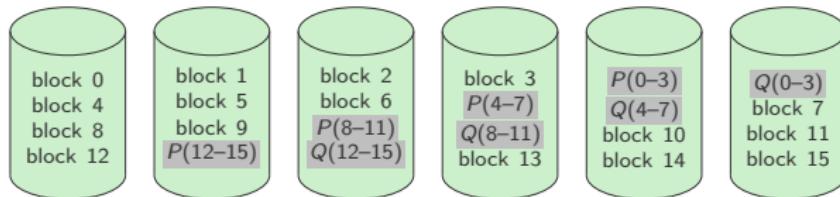
# RAID 5

- Accès indépendant avec **block-interleaved distributed parity**
  - Nécessite  $N + 1$  disques
- Données et parités réparties sur tous les disques
  - Tout en gardant données et parités sur des disques différents
  - Système le plus couramment utilisé
  - Évite la sur-utilisation d'un disque par rapport aux autres



# RAID 6

- Accès indépendant avec  $P + Q$  redundancy
  - Nécessite  $N + 2$  disques
- Stocke plus d'information de redondance que RAID 5
  - Peut résister au crash de plusieurs disques
  - Codes de Reed-Solomon à la place des parités



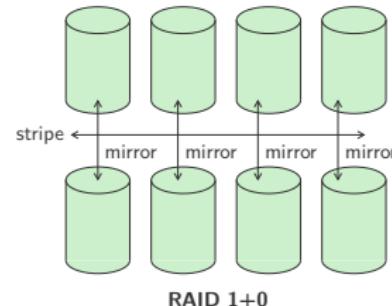
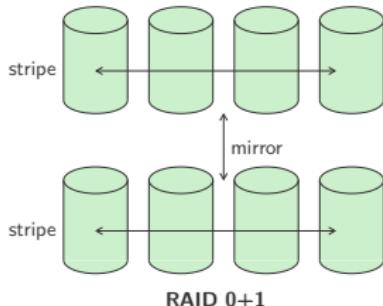
# RAID 0+1 / 1+0

- Combinaison de RAID 0 et RAID 1

*RAID 0 pour la performance et RAID 1 pour fiabilité*

- RAID 1+0 est **plus tolérant** aux pannes que RAID 0+1

- Deux RAID 0 du RAID 0+1 tombent si deux disques tombent
- Aussi, meilleures performances que RAID 5, mais plus coûteux



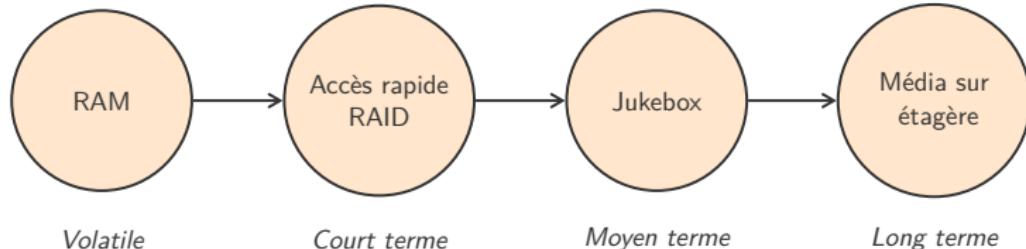
# Stockage hiérarchisé

- Plusieurs médias à disposition

*Hierarchical Storage Management (HSM)*

- Migration des données vers des systèmes moins performants

*Lorsque les données ne sont plus utilisées*





# Performance

# Performance de la cache

- Cache utilisée entre la mémoire principale et le CPU

*Création d'une mémoire interne à deux niveaux*

- Cache mémoire principale dans l'**architecture du système**

*Implémenté en hardware et invisible pour l'OS*

- **Autres mécanismes** de cache implémentés par l'OS

- Gestion de la mémoire virtuelle (TLB pour table des pages)
- Accès aux données du disque (buffer pour secteurs)

# Localité (1)

- Références mémoires regroupées en **clusters**

*Déplacement du cluster sur une longue période de temps*

- Principe de localité expliqué de **manière intuitive**

- Majorité d'un programme est séquentiel (sauf branch et call)
- Séquence d'appels et returns consécutifs rare
- Itération d'un petit nombre d'instructions répétées
- Traitement de structures de données (tableau, séquence...)

# Localité (2)

- Accès à des zones de mémoire proches par **localité spatiale**
  - Instructions à exécuter forment des clusters
  - Exécution d'instructions de manière séquentielle
  - Utilisation de large blocs de cache avec prefetching
- Accès régulier aux mêmes zones par **localité temporelle**
  - Accès à des zones mémoires récemment accédées
  - Répétition d'une séquence d'instructions dans une boucle
  - Mémoire cache pour instructions récemment exécutées

# Mémoire à deux niveaux

- Deux niveaux de mémoire pour exploiter la localité
  - $M_1$  de **haut-niveau** : petite, + rapide et + chère par bit
  - $M_2$  de **bas-niveau** : grande, – rapide et – chère par bit
- $M_1$  stockage temporaire de parties de  $M_2$ 
  - 1 Tentative de récupération de la donnée dans  $M_1$
  - 2 Si pas là, copie de la zone mémoire de  $M_2$  vers  $M_1$
  - 3 Nouvelle tentative dans  $M_1$  qui réussira donc

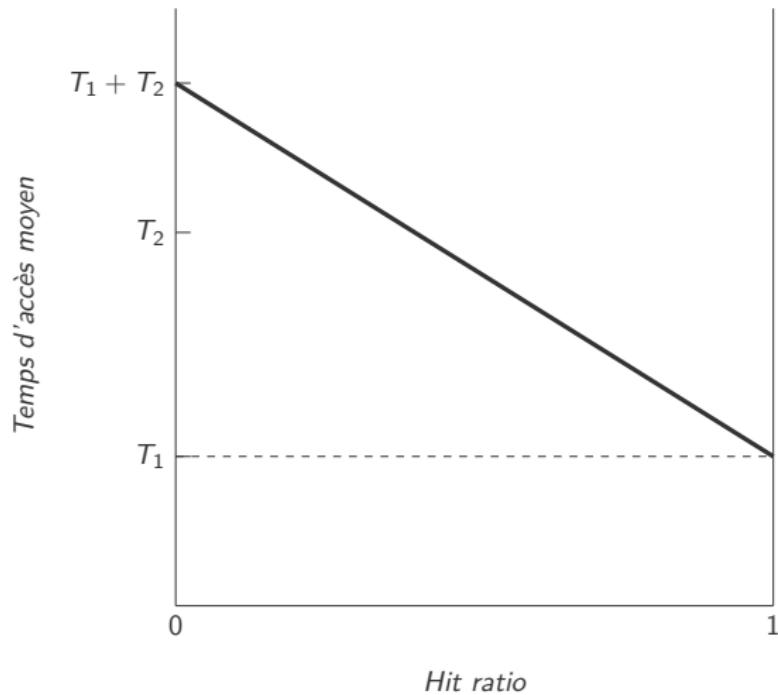
# Temps d'accès moyen (1)

$$\begin{aligned}T_s &= H \times T_1 + (1 - H) \times (T_1 + T_2) \\&= T_1 + (1 - H) \times T_2\end{aligned}$$



- Temps d'accès moyen  $T_s$  où
  - $T_1$  temps d'accès à  $M1$  (cache, cache de disque...)
  - $T_2$  temps d'accès à  $M2$  (mémoire principale, disque...)
  - $H$  hit ratio (fraction de temps où donnée trouvée dans  $M1$ )

## Temps d'accès moyen (2)



# Cout par bit moyen

$$C_s = \frac{C_1 S_1 + C_2 S_2}{S_1 + S_2}$$

- Cout par bit moyen  $C_s$  où
  - $C_1$  cout par bit moyen de  $M1$
  - $C_2$  cout par bit moyen de  $M2$
  - $S_1$  taille de  $M1$
  - $S_2$  taille de  $M2$
- Désir d'avoir  $C_s \approx C_2$  : donc  $S_1 \ll S_2$  en supposant  $C_1 \gg C_2$



# Efficacité d'accès (1)

- Désir d'avoir  $T_s \approx T_1$  : donc hit ratio très proche de 1

Puisque  $T_1 \gg T_2$  et donc  $T_s \gg T_1$



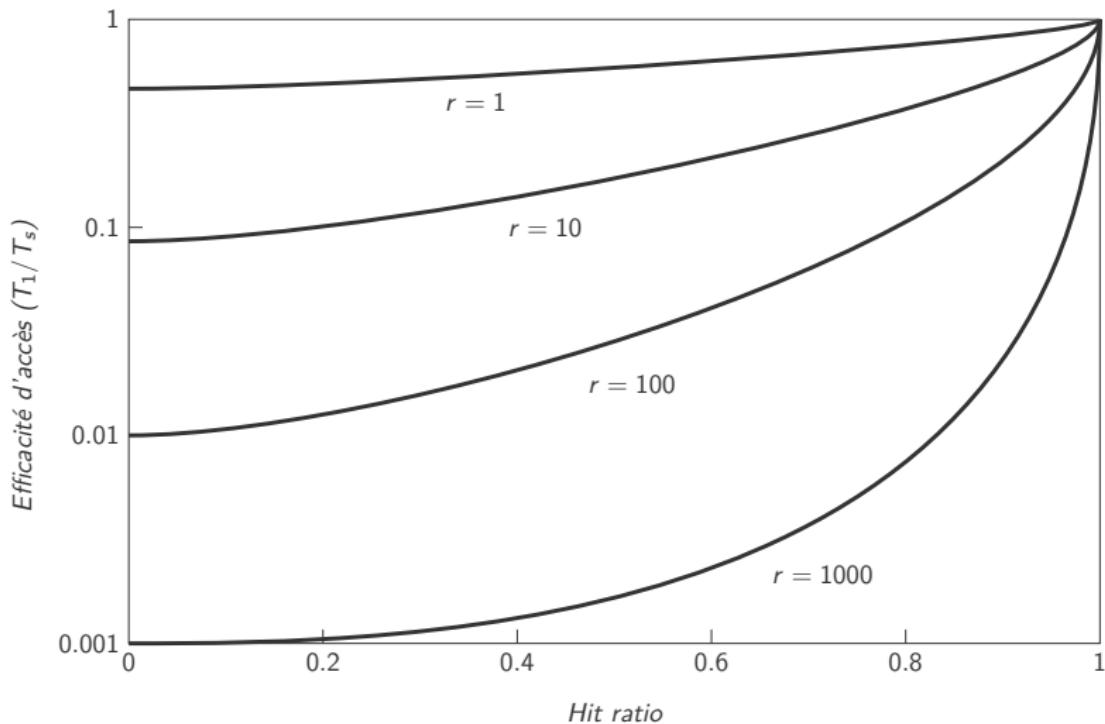
- Conditions pour choisir le meilleur  $M1$

- Minimiser le cout nécessite de diminuer  $S_1$
- Augmenter le hit ratio nécessite d'augmenter  $S_1$

- Efficacité d'accès mesure combien proche  $T_s$  est de  $T_1$

$$\frac{T_1}{T_s} = \frac{1}{1 + (1 - H)r} \quad \text{avec } r = \frac{T_2}{T_1}$$

## Efficacité d'accès (2)



# Degré de localité (1)

- Hit ratio toujours à 1 si **même tailles** ( $S_1 = S_2$ )

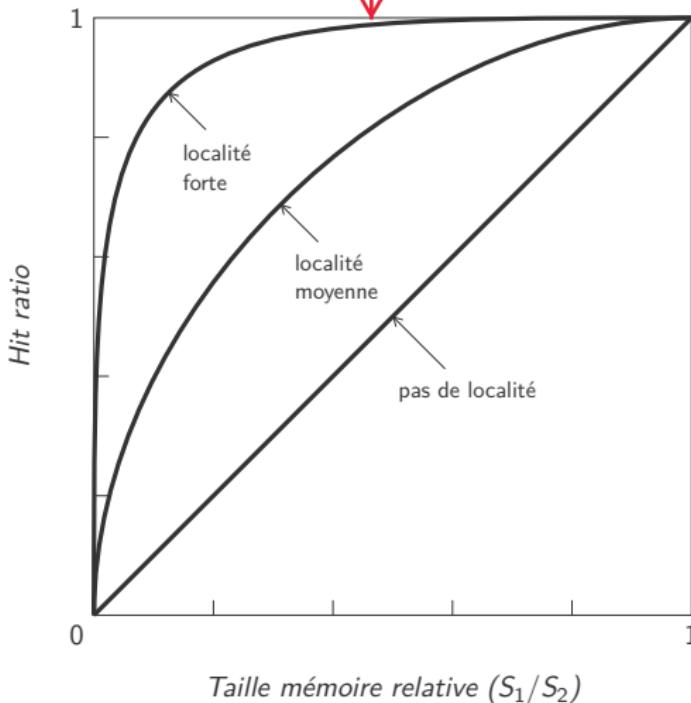
*Tous les éléments de  $M_2$  se retrouvent dans  $M_1$*

- Influence sur le hit ratio de la **présence de localité**

- Relation linéaire entre  $S_1/S_2$  et hit ratio sans localité
- Avec localité forte, bon hit ratio même avec  $S_1/S_2$  plus bas
- Hit ratio de 0.75 avec petit  $S_1$ , peu importe  $S_2$

## Degré de localité (2)

Point optimal

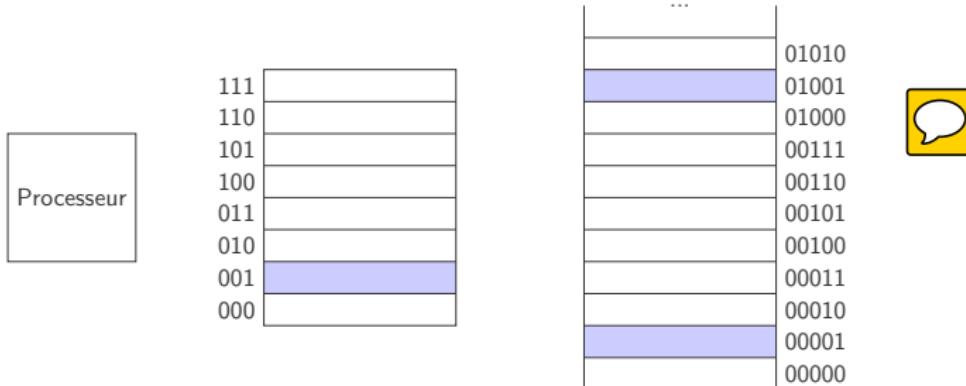


# Cache



# Cache de base

- Stockage de mots correspondants à des adresses mémoire
  - Le processeur fournit une adresse à la cache (hit ou miss)
  - En cas de miss, la cache va charger le mot depuis la mémoire
- Mapping direct entre adresse mémoire et bloc de cache  
 $addr \text{ (mod } nb\text{blocs)} + un \text{ tag (partie haute de } addr) + valid \text{ bit}$



# Lecture en cache



- Demande effectuée

1/ Adresse  $10110_2$  correspond à entrée  $110_2$  et donc MISS

- Contenu de la cache

V	Tag	Donnée
000	N	
001	N	
010	N	
011	N	
100	N	
101	N	
110	N	
111	N	

# Lecture en cache

- Demande effectuée

1/ Adresse  $10110_2$  correspond à entrée  $110_2$  et donc MISS

- Contenu de la cache

V	Tag	Donnée
000	N	
001	N	
010	N	
011	N	
100	N	
101	N	
110	Y	$10_2$ <i>Mem[10110<sub>2</sub>]</i>
111	N	

# Lecture en cache

- Demande effectuée

2/ Adresse  $11010_2$  correspond à entrée  $010_2$  et donc MISS

- Contenu de la cache

V	Tag	Donnée
000	N	
001	N	
010	N	
011	N	
100	N	
101	N	
110	Y	$10_2$ $Mem[10110_2]$
111	N	

# Lecture en cache

- Demande effectuée

2/ Adresse  $11010_2$  correspond à entrée  $010_2$  et donc MISS

- Contenu de la cache

V	Tag	Donnée
000	N	
001	N	
010	Y $11_2$	<i>Mem[11010<sub>2</sub>]</i>
011	N	
100	N	
101	N	
110	Y $10_2$	<i>Mem[10110<sub>2</sub>]</i>
111	N	

# Lecture en cache

- Demande effectuée

3/ Adresse  $10110_2$  correspond à entrée  $110_2$  et donc HIT

- Contenu de la cache

V	Tag	Donnée
000	N	
001	N	
010	Y	$11_2$ $Mem[11010_2]$
011	N	
100	N	
101	N	
110	Y	$10_2$ $Mem[10110_2]$
111	N	

# Lecture en cache

- Demande effectuée

4/ Adresse  $10010_2$  correspond à entrée  $010_2$  et donc MISS

- Contenu de la cache

V	Tag	Donnée
000	N	
001	N	
010	Y $11_2$	<i>Mem[11010<sub>2</sub>]</i>
011	N	
100	N	
101	N	
110	Y $10_2$	<i>Mem[10110<sub>2</sub>]</i>
111	N	

# Lecture en cache

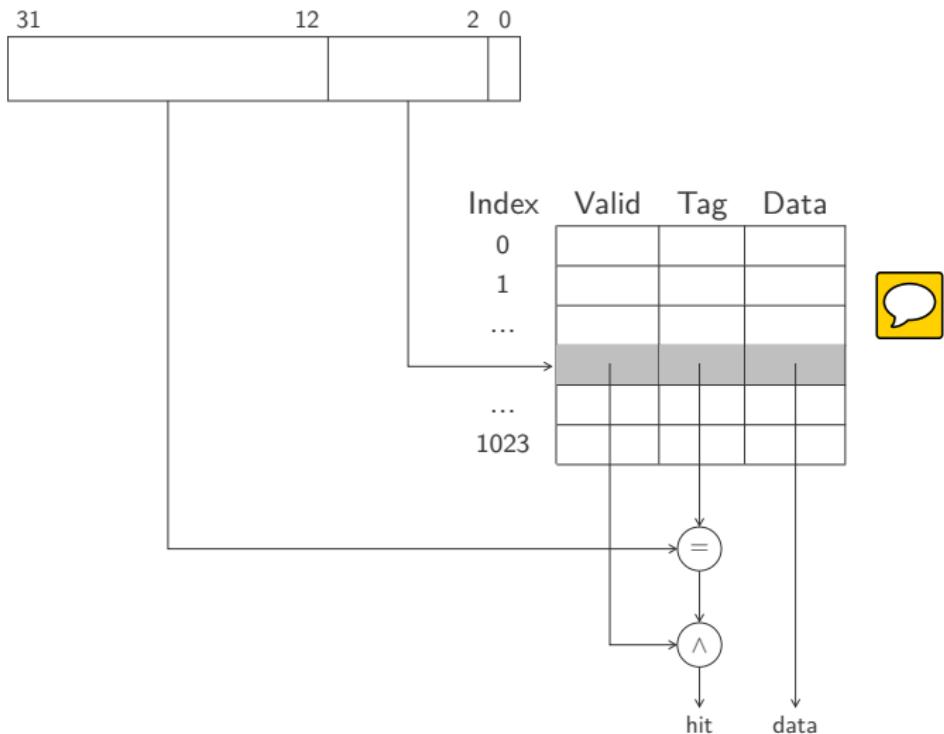
## ■ Demande effectuée

4/ Adresse  $10010_2$  correspond à entrée  $010_2$  et donc MISS

## ■ Contenu de la cache

V	Tag	Donnée	
000	N		
001	N		
010	Y $10_2$	$Mem[10010_2]$	
011	N		
100	N		
101	N		
110	Y $10_2$	$Mem[10110_2]$	
111	N		

# Accès mémoire cache



# Taille de la cache (1)

## ■ Données de départ

- Adresses sur 32 bits et cache de type mapping direct
- Taille de la cache fait  $2^n$  blocs, donc  $n$  bits pour l'index
- Taille des blocs fait  $2^m$  mots ( $2^{m+2}$  octets), donc  $m$  bits pour identifier le mot dans le bloc et 2 bits pour identifier l'octet dans le mot

## ■ Nombre de bits pour le **tag**

$$|tag| = 32 - (n + m + 2)$$

## ■ Nombre **total de bits** dans la cache

$$2^n \times (|bloc| + |tag| + |valid|)$$



# Taille de la cache (2)

- **Données de départ**
  - Taille de la cache fait 1024 blocs, donc 10 bits pour l'index
  - Taille des blocs fait 1 mot (4 octets = 32 bits), donc 0 bit pour identifier le mot dans le bloc et 2 bits pour identifier l'octet dans le mot
- Nombre de bits pour le **tag**
$$|tag| = 32 - (10 + 0 + 2) = 20$$

- Nombre **total de bits** dans la cache
$$2^{10} \times (32 + 20 + 1) = 54272 \text{ bits}$$
- En **excluant** tag et valid bit :  $2^{10} \times 32 = 32768 \text{ bits} = 4 \text{ Kio}$

# Écriture en cache

- Donnée **écrite en cache**, mais pas en mémoire principale

*Inconsistance entre la mémoire et la cache*

- **Write-through** écrit en cache et en mémoire

*En cas de MISS, overhead d'opérations mémoire*

- **Write-back** écrit en cache uniquement

*Écriture en mémoire principale lors d'un remplacement en cache*

# Crédits

- <https://www.flickr.com/photos/baggis/3576064503>
- <https://www.flickr.com/photos/materod/4809628635>
- <https://www.flickr.com/photos/marksze/4215239694>
- <https://www.flickr.com/photos/126080172@N03/14714849808>
- <https://www.flickr.com/photos/30887652@N08/2978460913>
- <https://www.flickr.com/photos/16111172@N03/1744666123>
- <https://www.flickr.com/photos/samsungtomorrow/8018119985>
- <https://www.flickr.com/photos/andrewcurrie/4542687657>
- [https://www.flickr.com/photos/ibm\\_media/7203083302](https://www.flickr.com/photos/ibm_media/7203083302)
- [https://www.flickr.com/photos/ama\\_nita/4606994230](https://www.flickr.com/photos/ama_nita/4606994230)
- <https://www.flickr.com/photos/dmott9/5662174145>