

**INF 227**

***Hierarchie de memoires***

**B. Dupouy**

***Plan***

- Definition
- Principe de localite
- Hit et miss
- Gestion d'une hierarchie

## Hierarchies de memoires

---

## Hierarchies de memoires

---

### *Introduction*

- Ce que souhaitent les utilisateurs :
  - Memoire de capacite infinie,
  - Temps d'accès très court (pour les données et les instructions), c'est à dire :  
temps d'accès à la memoire = temps de cycle processeur
- Temps d'accès :
  - Bande passante,
  - Latence,

## Hierarchies de memoires

---

## Hierarchies de memoires

---

### ***Principe de localite***

- Les accés memoire peuvent présenter deux types de propriétés :
  - localité temporelle : une adresse qui vient d'être référencée le sera sans doute à nouveau dans un futur proche,
  - localité dans l'espace : si une adresse vient d'être référencée, ses voisines le seront également dans un futur proche.
- *Working set* (espace de travail) de fenêtre  $\Delta$  au temps  $t$ , ou  $W(t, \Delta)$  :  
ensemble des adresses référencées entre  $t$  et  $t+\Delta$

## Hierarchies de memoires

---

## Hierarchies de memoires

---

### *Exemple de localite*

- Localite temporelle (ici, sur les instructions) :

```
for (i=0; i <DIM; i++) S = S + Tab[i];
```

- Localite spatiale (ici, sur les donnees) :

```
for (i=0; i <DIM; i++) Buff[i] = Tab[i];
```

- Attention au parcours des tableaux :

```
for (i=0; i <DIM; i++) Tab[i][j] = 0;
```

rangement ligne / colonne ou colonne / ligne ‡ principe de localite respecte ou non.

## Hierarchies de memoires

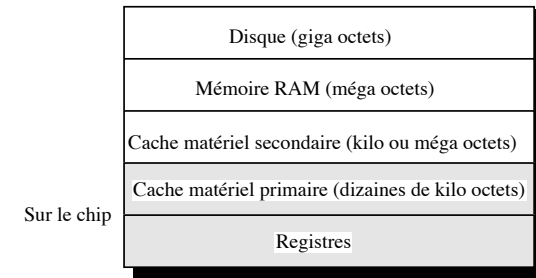
---

## Hierarchies de memoires

---

### *Hierarchie (1)*

- Du plus rapide et plus petit (niveaux bas) au plus lent et plus volumineux (niveaux hauts) :



- Quand la capacite decroit, le cout du bit augmente.
- Hierarchie logicielle / materielle

## Hierarchies de memoires

---

## Hierarchies de memoires

---

### *Hierarchie (2)*

- Objectifs de la gestion de la hierarchie de memoires :
  - les niveaux bas (rapides) gèrent la majorité des références memoire, ils gardent les adresses les plus récemment utilisées (localité temporelle).
  - utiliser la localité spatiale pour les transferts entre niveaux, qui sont faits par blocs.
- Les performances dependent :
  - de la structure et du fonctionnement de la hierarchie
  - des propriétés de localité du PROGRAMME.

## Hierarchies de memoires

---

## Hierarchies de memoires

---

### *Hit et miss*

- Soit une hierarchie a deux niveaux  $M_1$  et  $M_2$  tels que:

	Taille	Temps d'accès	Coût/bit
$M_1$	$S_1$	$T_1$	$C_1$
$M_2$	$S_2$ , avec $(S_1 < S_2)$	$T_2$ , $(T_1 < T_2)$	$C_2$ , avec $(C_1 > C_2)$

- Définitions :

<i>hit</i>	référence mémoire satisfaite par $M_1$
<i>miss</i>	référence mémoire effectuée par $M_2$
<i>hit ratio (h)</i>	Nbre de hits / Nbre total de références
<i>miss ratio (1-h)</i>	Nbre de misses / Nbre total de références

$T_1(T_2)$  = temps pour satisfaire un hit (miss).

## Hierarchies de memoires

---

## Hierarchies de memoires

---

### Performances

- Temps d'accès moyen à la mémoire :

$$T_m = h T_1 + (1-h) T_2$$

- Accélération S (speedup) due à  $M_1$ :

$$S = \text{Erreur !} = \text{Erreur !} = \text{Erreur !}$$

$$S = \text{Erreur !}$$

- Or :  $\text{Erreur !} < 1 \nmid$  si  $h = \text{Erreur !}$ , l'accélération est au plus de 2...

- si  $\text{Erreur !} = 10$ , une baisse de 1% sur un *hit ratio* voisin de 1 se traduit par une augmentation  $\approx 10\%$  du temps d'accès moyen.

## Hierarchies de memoires

---

## Hierarchies de memoires

---

### ***Gestion de la hierarchie***

- Soit une hierarchie à 2 niveaux :  $M_1$  et  $M_2$ .
  - quand charger un bloc de  $M_2$  dans  $M_1$  ?
  - où placer le bloc dans  $M_1$  ?
  - comment trouver une adresse dans  $M_1$  ?
  - quand  $M_1$  est plein, quel bloc remettre dans  $M_2$  ?
  - que faire en cas d'écriture ?
- Réponses :
  - lecture anticipée (*read ahead*) ou non,
  - cache purement associatif ou non
  - algorithme LRU,
  - écriture immédiate ou différée (*write back, write through*),

## Hierarchies de memoires

---

## Hierarchies de memoires

---

### ***Problemes de gestion***

- coherence entre la copie dans le niveau inferieur et l'element original :
  - commande `sync` de Unix
  - gestion write back ou write through des caches materiels,
  
- copies multiples d'un element et ecriture dans une de ces copies (caches sur les machines multiprocesseur, caches de `nfs`) :
  - propagation des copies
  - invalidation des copies