Gestion de la mémoire

Olivier Dalle
Université de Nice - Sophia Antipolis
http://deptinfo.unice.fr/

9. Gestion de la Mémoire

1. **Introduction**
2. Gestionnaire de mémoire
3. Gestion par segmentation
4. Gestion par pagination
5. Algorithmes de pagination
6. Répartition des cadres entre processus

Exposé du Problème (Plan)

- Mémoire physique = ressource physique
  - Rare
  - Demandée par tous les processus
- Quantité de mémoire physique insuffisante
  - Extension par de la mémoire disque
  - **Mémoire virtuelle**
    - Principe
    - mise(s) en œuvre possible(s)
  - Stratégies du partage de la mémoire
    - swap de processus ou de parties de processus

O. Dalle

3

Différents types de Mémoire

- Mémoire volatile
  - Constituée de circuits intégrés
    - accès rapide, mais coût élevé
  - Exemples (performance décroissante)
    - Registres CPU
    - Mémoire associative
      - Mémoire tampon = *cache* entre les registres et la RAM
    - Mémoire centrale, constituée de *RAM*
      - RAM = Random Access Memory : chaque mot mémoire est directement accessible par son adresse

O. Dalle

4

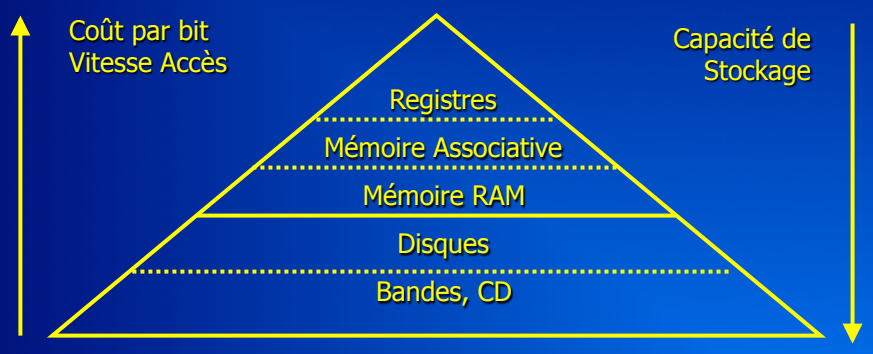
Différents types de Mémoire (2)

- Mémoire permanente (secondaire, auxiliaire)
  - Constituée de supports de stockage massifs
    - magnétiques (disque durs, bandes)
    - optiques, magnéto-optiques (CD, DVD, …)
    - électroniques persistants (EEPROM/flash)
  - Accès plus lent, mais coût plus faible (généralement)

O. Dalle

5

Hiérarchie des Mémoires



O. Dalle

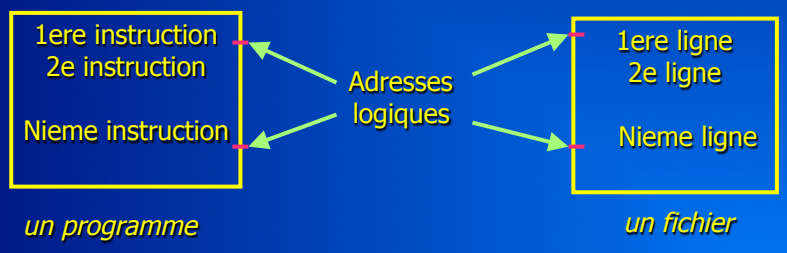
6

9. Gestion de la Mémoire

1. Introduction
2. **Gestionnaire de mémoire**
3. Gestion par segmentation
4. Gestion par pagination
5. Algorithmes de pagination
6. Répartition des cadres entre processus

Modèles de représentation de l'information

- Modèle Logique : vision de l'utilisateur
  - Généralement séquentielle : Information = suite d'octets



Modèles de représentation de l'information (2)

- Modèle physique : stockage réel dans la machine
  - Informations pas forcément contiguës !
  - Plusieurs représentations physiques peuvent exister pour une même représentation logique

    Exemple:

    Un programme en cours d'exécution existe en général sous deux formes physiques :

    1. Stocké sur le disque dur
    2. Chargé en mémoire lors de son exécution

O. Dalle

9

Modèles de représentation de l'information (3)

- Exemple de représentations physiques associées à une représentation logique : exécutable
  - Sous forme logique = suite d'adresses mémoire

    …

    Ajouter 10 à la valeur contenue à l'adresse mémoire 50

    …
  - Sous forme de fichier = blocs « éparpillés » sur le disque

  - Sous forme de programme en mémoire = une ou plusieurs zones (allouées et changeant dynamiquement)

O. Dalle

10

Rôles d'un gestionnaire de mémoire

- Assurer la correspondance
  - entre ***la*** représentation logique et ***les*** représentations physiques
  - de façon **transparente**
- Pallier au manque de RAM
  - Code et données de + en + volumineux
  - Utilisation de **mémoire virtuelle**
    - Représentation **logique** de taille plus grande que la mémoire centrale
    - Rem : au moins une représentation physique quelque part !

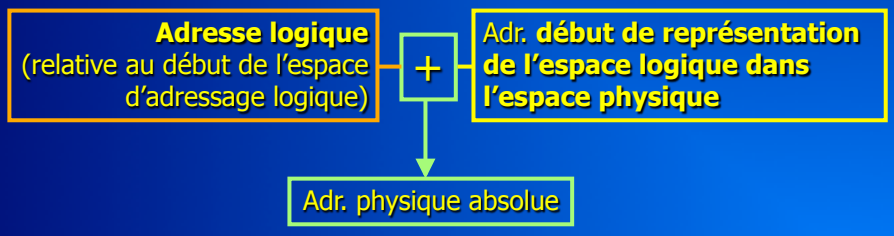
O. Dalle

11

Implantation de la Correspondance LOGIQUE/PHYSIQUE

- Se fait au niveau matériel pour être efficace
  - Memory Management Unit (MMU)
  - Mémoire Associative

Exemple le plus rudimentaire :

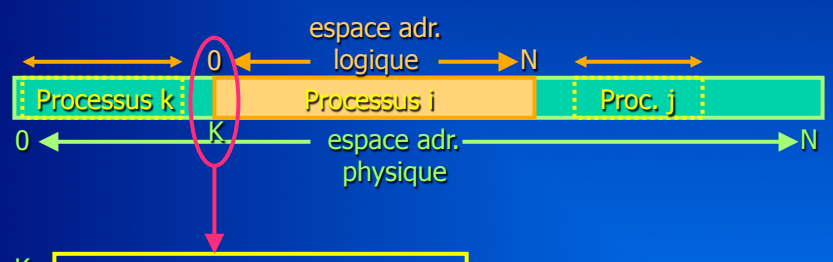


O. Dalle

12

Implantation de la Correspondance LOGIQUE/PHYSIQUE (2)

Exemple le plus rudimentaire (illustration)



K : Adr. **début de représentation de l'espace logique (de Pi) dans l'espace physique**

O. Dalle

13

Mise en Œuvre de la Mémoire Virtuelle

- Constatation : une instruction CPU ne nécessite pas plus de 8 accès mémoire
  - ⇒ Pas nécessaire que **toute** la représentation logique soit constamment en mémoire
- Actuellement sans défauts majeurs
  - Totalement transparente pour le programmeur
  - Efficace du point de vue du S.E.
  - Permet la cohabitation de plusieurs processus **prêts** en mémoire centrale

O. Dalle

14

Classification des Gestionnaires

- Selon découpage de la représentation logique (souvent guidé par l'architecture du CPU)
  - Découpage en **partitions de taille fixe**

    Une partition = une **page**

    => **gestionnaire par pagination**
    - Exemple de CPU : Sparc, 680XX (motorola), PDP11 (dec)
    - Exemple de S.E. : Unix BSD, Windows NT

O. Dalle

15

Classification des Gestionnaires (2)

- Découpage en **partitions de taille variable**

  Une partition = un **segment**

  => **gestionnaire par segmentation**
  - Exemple de processeur : 8086 -> 80286 (intel)
  - Exemple de SE : MSDOS
- Combinaison des 2 découpages (segment + page)

  => **gestionnaire par segmentation paginée**
  - Exemples de processeurs : 80386 -> Pentiums (intel)
  - Exemples de SE : Linux, Multics, OS/2

O. Dalle

16

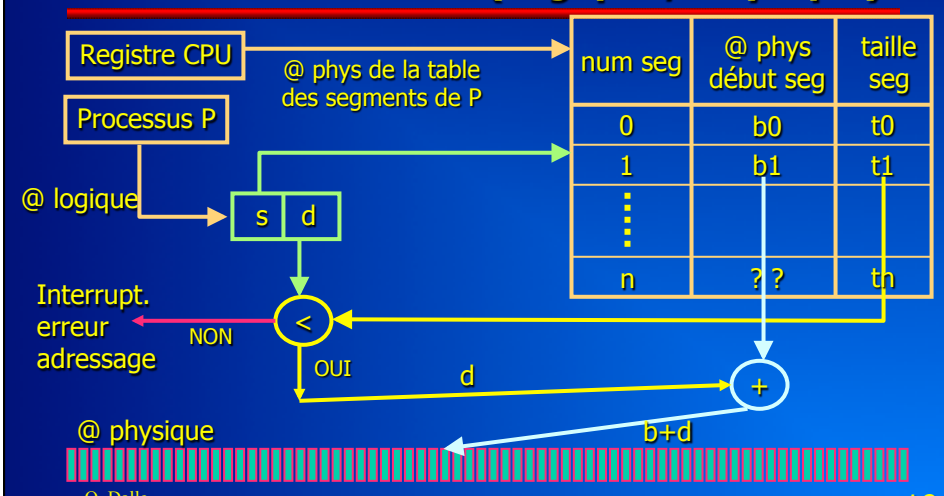
9. Gestion de la Mémoire

1. Introduction
2. Gestionnaire de mémoire
3. **Gestion par segmentation**
4. Gestion par pagination
5. Algorithmes de pagination
6. Répartition des cadres entre processus

Fonctionnement

- Hypothèses : le compilateur a généré un exécutable constitué de segments
  - Taille d'un segment << taille MC
- Un segment = un numéro + une taille
- Une adresse logique = (s,d)
  - s = numéro du segment
  - d = déplacement dans le segment (*offset*)
- Chaque processus a une table des segments
  - Doit être en MC lorsque le processus est **prêt**
  - Sinon, elle peut être stockée en zone de swap (sur le disque)

Schéma de Conversion des Adresses (Logique / Physique)



O. Dalle

19

Allocation des segments en Mémoire Centrale

- Le segment s n'est pas chargé en MC (?? dans @ phys début de seg)
  - ⇒ **Déroutement** « défaut de segment »

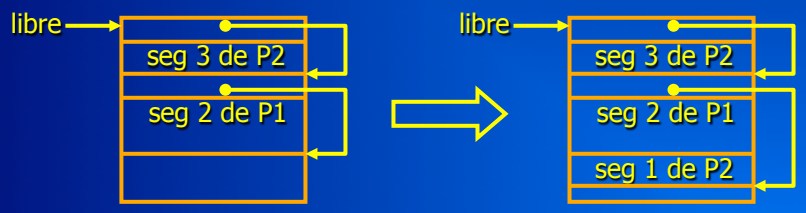
Il faut trouver une succession de mots libres en MC de longueur >= taille seg

O. Dalle

20

Allocation des segments en Mémoire Centrale (2)

- Le gestionnaire maintient une liste de blocs mémoire libres/occupés



Allocation des segments en Mémoire Centrale (3)

- Problème de la segmentation :
  - Apparition du phénomène de **fragmentation** de la mémoire centrale
    - Fragmentation **externe** (entre segments)
    - Situation extrême : la somme des blocs libres serait suffisantes pour héberger un segment, mais ils ne sont pas contigus.

Politiques de Choix d'un bloc de Mémoire Libre (pour charger un segment)

- **First Fit** : le premier qui est assez grand.
  - Avantage : choix rapide
- **Best Fit** : meilleur ajustement (taille la plus proche possible du segment à héberger)
  - Avantage : fragmentation externe importante mais fragment de petite taille
  - Inconvénients : choix lent
- **Worst Fit** : le plus grand résidu
  - Avantage : fragmentation externe faible (statistiquement) car résidu a de bonne chance de pouvoir reservir
  - Inconvénient : choix lent

Que faire si aucun bloc libre ne convient ?

- **Compacter** la Mémoire Centrale
  - Cher et lent
  - Plutôt réservé aux exécutifs de systèmes ou langages (car tout est bloqué)
- **Différer** l'accès
  - Swap out : décharger le processus depuis la mémoire vers le disque (en attendant que ça aille mieux)
  - Swap in : recharger le processus depuis le disque (lorsque ça va – peut-être – mieux)
    - Stratégie de chargement confié à l'algorithme de « swapping » (souvent implémenté sous forme d'un processus démon : le « swapper »)

Que faire si aucun bloc libre ne convient ? (2)

- **Récupérer** de force de la place
  - Déchargement vers la zone disque
    - D'un segment
    - D'un processus entier
  - Question : lequel choisir ?
    - Cf. politiques étudiées dans le cadre de la pagination…
- Quand bloc libre est (enfin) trouvé :
  - Mise à jour de la table des segments du processus
  - Re-conversion de l'adresse logique en adresse physique

O. Dalle

25

Avantages de la Gestion Segmentée

- **Partage** et **Protection** aisée des segments : un segment est une entité logique
  - On peut lui attacher des propriétés
    - mode d'accès (protection Read/Write/eXécute)
    - copie « paresseuse » : on évite de faire une copie d'un segment tant qu'un processus ne modifie pas son contenu (*Copy On Write*)
    - verrouillage en mémoire

O. Dalle

26

Avantages de la Gestion Segmentée

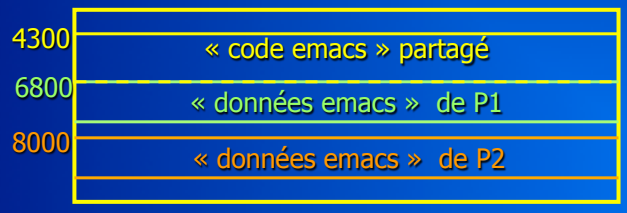
- Exemple de **partage** et **protection**

Table des segments de P1

num seg	RWX	@ phys début seg	taille seg
0	RX	4300	2500
1	RW	6800	400

Table des segments de P2

num seg	RWX	@ phys début seg	taille seg
0	RX	4300	2500
1	RW	8000	600



9. Gestion de la Mémoire

1. Introduction
2. Gestionnaire de mémoire
3. Gestion par segmentation
4. **Gestion par pagination**
5. Algorithmes de pagination
6. Répartition des cadres entre processus

Fonctionnement

- La mémoire centrale est découpée en **cadres** (*frames*)
  - Taille constante **c**
    - Exemple linux : **c = 4 Ko**
- Espace logique est découpé en **pages**
  - Taille constante **c**
    - la même que celle des cadres : les pages sont placées dans des cadres

O. Dalle

29

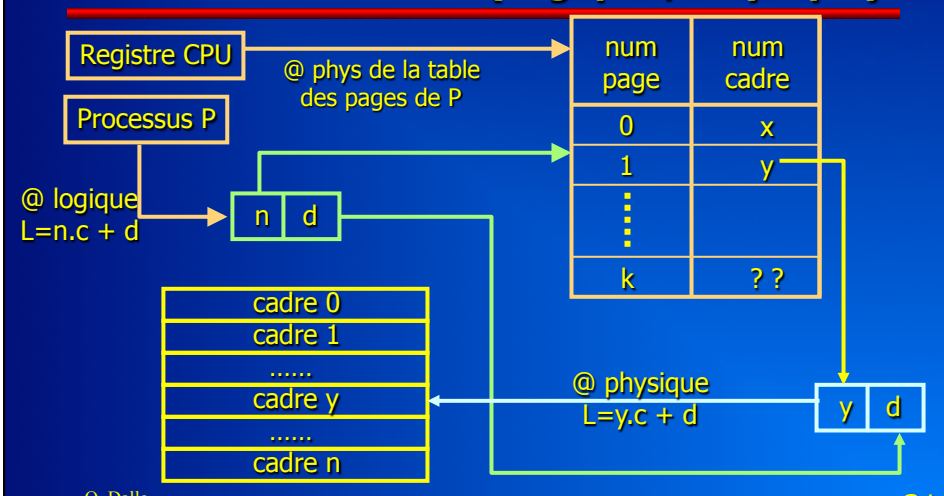
Fonctionnement (2)

- Le compilateur n'a pas de rôle spécial
  - Il engendre un espace d'adressage logique linéaire
  - La conversion d'une adresse logique L sous la forme L=(n,d) est **implicite**
    - c = taille de page
    - **n** = L div c : numéro de la page
    - **d** = L mod c : déplacement dans la page
    - ⇒ L = **n**.c + **d**

O. Dalle

30

Schéma de Conversion des Adresses (Logique / Physique)



O. Dalle

31

Application Pratique
(cf Tanenbaum, p 104)

- H1 : on dispose d'un CPU 16 bits
  - les registres peuvent recevoir des adresses sur 16 bits
  - L'espace d'adressage du processus est de 2^16= 64 Ko
- H2 : Le MMU impose une taille de cadres et pages de 4 Ko (2^12)
  - Le nombre max de pages pour l'espace d'adressage d'un processus = 2^16/2^12=16

O. Dalle

32

Application Pratique (2)

- H3 : La table de pages du processus est la suivante :
- Questions
  1. Dans quel cadre se trouve l'adresse logique 8196 ?
  2. A quel adresse physique correspond l'adresse logique 8196 ?

page	cadre/disk	pres.
0	010	1
1	001	1
2	110	1
3	000	1
4	100	1
5	011	1
6	706	0
…		
14	035	0
15	965	0

O. Dalle

33

Application Pratique (3)

- Réponse :
  - 8196 = 2^13 + 2^2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0

num page virtuelle

déplacement dans la page

page	cadre/disk	pres.
0	010	1
1	001	1
2	110	1
3	000	1
4	100	1
5	011	1
6	706	0
…		
14	035	0
15	965	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0

adresse physique = 24580

O. Dalle

34

Allocation des pages en Mémoire Centrale

- Page absente : déroutement de type « défaut de page »
  - Trouver un cadre libre
    - N'importe lequel convient (pas de fragmentation externe)
    - Pas de cadre libre : cf algorithme de pagination …
  - Charger la page depuis l'endroit où elle est stockée sur le disque
    - Dans le swap
    - Directement depuis l'exécutable
  - Problème de fragmentation : uniquement **interne** au pages
    - Seulement dans la dernière page
    - En moyenne, ½ page perdue

O. Dalle

35

Partage et Protection des pages

- Plus difficile qu'en gestion segmentée
  - Découpage en page arbitraire
    - une page peut mélanger du code et des données
  - Plusieurs pages peuvent être nécessaires pour stocker une même information
    - Les informations de partage et de protection doivent être dupliquées pour chaque page

O. Dalle

36

9. Gestion de la Mémoire

1. Introduction
2. Gestionnaire de mémoire
3. Gestion par segmentation
4. Gestion par pagination
5. **Algorithmes de pagination**
6. Répartition des cadres entre processus

Généralités

- Problème : choisir un cadre à libérer pour le charger avec une nouvelle page
  - Peut exiger la mise à jour d'une page modifiée (cf. bit de modification)
- Stratégies similaires pour décharger un segment
  - Avec contrainte supplémentaire de trouver une zone de mémoire contiguë pour le segment
- Critère d'évaluation d'une stratégie de pagination

$$\text{TauxDéfautsDePages} = \frac{\text{NombreChargementsDePages}}{\text{NombreDeRéférences}}$$

  - Exemple : 100 référence successives en memoire exigent 10 chargements de page => TauxDefautsDePage=10%

Algorithme Optimal (OPT)

- L'algorithme choisit de remplacer la page qui sera référencée **le plus tard dans le futur**
- Problème : on ne connaît pas le futur !
  - Intérêt purement théorique, afin de comparer avec d'autres algorithmes
    - Un algorithme est d'autant meilleur qu'il est proche de la limite théorique obtenue avec OPT

O. Dalle

39

Algorithme FIFO

- L'algorithme choisit de remplacer la page qui est la plus ancienne a avoir été chargée
- Problème : il peut s'agir d'une page qui est utilisée en permanence
  - Exemple : la partie de l'interpréteur shell zsh qui affiche le prompt …
  - Perte de temps, car il faut recharger immédiatement la page

O. Dalle

40

Algorithme FIFO de la 2nde Chance

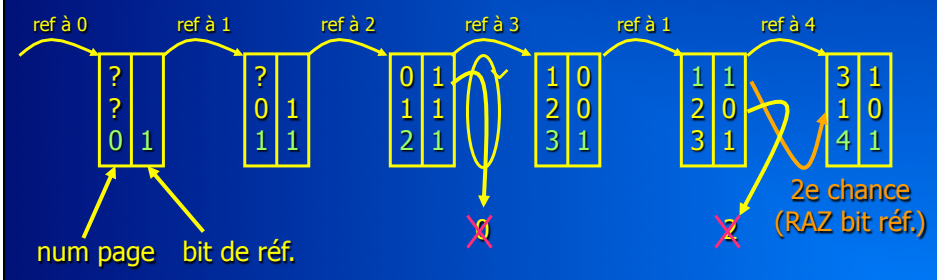
- L'algorithme choisit de remplacer la page la plus ancienne à avoir été chargée, après lui avoir donné une 2nde chance (de rester en mémoire)
- Détail :
  - Une page référencée à son bit de référence qui passe à 1
  - Si bit de référence à 1, l'algo donne une seconde chance
    - Remise a Zéro du bit de réf
    - Déplacement de la page en queue de FIFO (comme si elle venait d'être chargée)
  - L'algo choisit comme « victime », la première page avec bit de réf. à 0

O. Dalle

41

Algorithme FIFO de la 2nde Chance (2)

- Exemple :
  - Soit la suite de réf. aux pages
    0 1 2 3 1 4
  - Nb cadres disponibles = 3



O. Dalle

42

Algorithme FIFO de la 2nde Chance (3)

- Remarque :

  Cet algo se comporte exactement comme FIFO quand tous les bits de réf sont à 1
- Implantations
  - Déplacer une page est coûteux => on évite
    - Algorithme de pagination de l'horloge (simule une aiguille qui fait le tour de la FIFO)

O. Dalle

43

Algorithme Least Recently Used (LRU)

- Remplace la page qui a été le moins récemment utilisée
  - S'appuie sur le **principe de localité**
    - **Dans le temps**

      Exemple : dans une boucle, les variables qui viennent d'être accédées le seront à nouveau très prochainement
    - **Dans l'espace**

      Exemple : lors du parcours d'un tableau, les références portent sur des adresses voisines

      Contre-exemple : parcours d'une liste construite dynamiquement…

O. Dalle

44

Algorithme Least Recently Used (LRU) (2)

- Implantation coûteuse
  - Il faut garder la date de référence de chaque page
  - Il faut rechercher la date la plus ancienne
- Mais performance proche de OPT

9. Gestion de la Mémoire

1. Introduction
2. Gestionnaire de mémoire
3. Gestion par segmentation
4. Gestion par pagination
5. Algorithmes de pagination
6. **Répartition des cadres entre processus**

Allocation égale

- Si M cadres et P processus
  - on réserve M/P cadres à chaque processus
  - Pb : P varie …

O. Dalle

47

Allocation Proportionnelle

- Nombre de cadres dépend de la taille de l'espace d'adressage du processus
  - Soit N, le nombre total de pages $N = \sum_{i=1}^{nbprocessus} N_i$
  - Alors le nombre de cadre *Mi* réservé à chaque Pi est $M_i = \frac{N_i \times M}{N}$
  - Exemple : M = 62 cadres, N= 137 pages
    P1 a 10 pages, P2 a 127 pages
    Alors, M1 = 4 cadres et M2 = 57 cadres

O. Dalle

48

Allocation globale (variable)

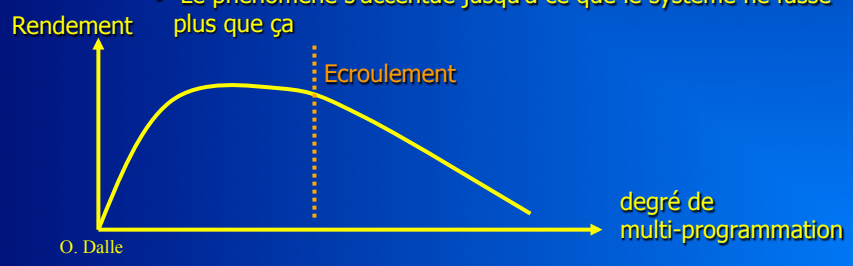
- Stratégie de répartition globale
  - Après une réservation initiale, si un processus n'utilise pas encore un cadre qui lui a été réservé, il peut être alloué à un autre processus

O. Dalle

49

Phénomène d'écroulement (trashing)

- Beaucoup de processus en mémoire
  - Nombreux défauts de page
    - Pendant le traitement des défauts de page, le rendement du système diminue
    - Le swapper estime qu'il n'y a pas assez de processus prêts et en charge de nouveaux (swap in)
    - Le phénomène s'accentue jusqu'à ce que le système ne fasse plus que ça



O. Dalle

50

Phénomène d'écroulement (trashing)

- Solutions
  - Eviter que le nombre de cadres alloué ne descende en dessous d'un certain seuil
    - Calculé en fonction du degré de localité des programmes
    - Utilisation de la notion d'espace de travail (on se base sur le nombre de pages distinctes utilisées récemment)
  - Si impossible : décharger le processus (swap out)
    - Le processus n'est plus éligible
    - Ses cadres sont attribués à un autre processus
    - Remarque : lors du rechargement, il est préférable que le swapper anticipe (préchargement)