Memoire Virtuelle

Comment exécuter certains programmes qui ont une taille supérieure à la taille de la mémoire?

Avec la mémoire virtuelle on peut l'exécuter (notamment au travers de la pagination, donc en chargeant en mémoire seulement les pages en exécution. Dès qu'elle est finie, on la renvoie, puis on charge les autres).

- Pour un processus, la mémoire virtuelle est le support de l'ensemble des informations potentiellement accessibles.
- C'est donc plus précisément, l'ensemble des emplacements dont l'adresse peut être engendrée par le processeur.
- L'allocation de mémoire consiste à concrétiser cette mémoire virtuelle par des supports physiques d'information tels que mémoire principale (RAM), disques magnétiques, etc. L'objectif est de concrétiser cette mémoire virtuelle via des supports physiques d'information. Comment le faire?

Donc le principe de la mémoire virtuelle est le suivant : *la taille de l'ensemble formé par le programme, les données et la pile peut dépasser la capacité disponible de la mémoire physique.*

Le système d'exploitation conserve les parties de programme en cours d'utilisation dans la mémoire principale, et le reste sur le disque.
Exemple:

- Un programme de 16 Mo peut s'exécuter sur une machine de 4 Mo de mémoire si les 4 Mo à garder en mémoire à chaque instant sont choisis avec attention, et que des parties du programme passent du disque à la mémoire, à la demande. Mémoire virtuelle et E/S
- Quand un programme attend le chargement d'une partie de lui-même, il est en attente d'E/S et ne peut s'exécuter; le CPU peut par conséquent être donné à un autre processus, de la même façon que pour tout autre système multiprogrammé

Le chargement des pages etc.. implique des opérations d'entrées sorties. Cela nécessite l'utilisation du processeur, il y a donc des degrés d'entrées/sorties qu'il ne faut pas dépasser.

2 techniques pour implanter une mémoire virtuelle:
Dans les deux cas le manque en mémoire centrale est compensé par la mémoire secondaire. (mémoire externe, un disque ou autre) • Le principe des deux méthodes consiste à ne maintenir en mémoire centrale que les instructions et les données nécessaires à l'exécution d'un programme à un instant t.

Overlays ou segments de recouvrement:

- Division du programmen en parties, appelées segments.
- Segment 0 exécuté en premier, puis il appele un autre segment. Les segments sont conservés sur le disque et chargés en mémoire par le système d'exploitation. Difficulté: Quelles sont les parties du programme qui doivent être stockées en mémoire à l'instant *t*, et lesquelles doivent être exécutées?

Voir exemple sur la diapo 65

Pagination à la demande:

- Mémoire virtuelle de taille bcp + grande que la mémoire phys.
- Il faut alors mettre en mémoire physique uniquement les pages de mémoire virtuelle dont les processus ont besoin pour leur exécution courante, les autres étant conservées sur mémoire secondaire (disque).
- Lorsqu'un processus accède à une page qui n'est pas en mémoire physique, l'instruction est interrompue, et un déroutement au système est exécutée. On dit qu'il y a défaut de page. (Interruption, + sauvegarde du contexte).
- Le système doit alors charger cette page, cette méthode qui consiste à charger une page uniquement lorsqu'elle est demandée s'appelle **pagination à la demande (demand paging).**

Si la mémoire est pleine et qu'on doit charger des pages, il faut remplacer celles existantes par les nouvelles.

- Dans un système paginé, les adresses générées par un programme sont appelées des adresses virtuelles et elles forment l'espace d'adressage virtuel. Les tables de page se trouvent au niveau de l'Unité de Gestion Mémoire.
- Ces adresses virtuelles ne vont pas directement sur le bus mémoire mais dans une unité de gestion mémoire (MMU, Memory Management Unit) qui fait correspondre les adresses virtuelles à des adresses physiques, en se basant sur une table de page.

La taille d'une case mémoire Réelle est la même taille qu'une page de l'espace Virtuel.

Voir l'exemple sur le diapo 70.

Pour voir si une page virtuelle a une case réelle en mémoire, on a un bit de présence/absence.

Défaut de page:
Si le programme essaye de faire appel à une page qui n'est pas présente? MMU remarque que page absente et fait procéder le CPU a un déroutement => s'appelle un *défaut de page*.

Voir exemple diapo 73.

Comment choisir la page victime? (page qui va être swap out).

Algorithmes de remplacement de page

Remplacement de page (local / global)

• Lors d'un défaut de page, s'il n'y a pas de case libre pour charger la page demandée comment choisir la page à remplacer ? Un algorithme de remplacement peut être :

- Local : si sa victime est choisie parmi les pages allouées au programme
- Global : si sa victime est choisie parmi l'ensemble de toutes les pages présentes en mémoire centrale.

Chaines de référence

• On appelle chaîne de références la séquence des numéros de pages référencées par un programme durant une exécution. • Un bon algorithme de remplacement doit diminuer le nombre de défauts de pages engendrés par une chaîne de référence définie par l'exécution d'un processus.

Problème: Le processeur ne sait q=pas les instructions qui vont être exécutées dans le futur.Or, on veut remplacer les pages qui ne feront l'objet d'aucune référence ultérieure, ou le plus tardivement possible.

L'algorithme optimal à ce but, mais il nécessite de connaître les chaînes de références. Or cen 'est pas possible.

Algorithme du Tirage Aléatoire:

Prends une page au pif. Sert surtout d'algo de comparaison.

Ordre chronologique de Chargement (FIFO):

- Il faut un compteur pour savoir quand est-ce que la page a été chargée (mettre dans une file FIFO, les nums des cases ou sont chargées les pages successives).
- Victime = page la plus ancienne
- Performances pas souvent bonnes.

Diapo 78:

On charge la page 7 dans un premier temps. Puis on veut la page 0, mais elle n'y est pas, donc il y a un défaut de page.

Ordre Chronologique d'utilisation (LRU: Least Recently Used):

• Cet algorithme tente d'approcher l'algorithme optimal, en utilisant la propriété de localité.
• Son principe s'explique comme suit: puisque les pages récemment utilisées ont une probabilité plus élevée que d'autres d'être utilisées dans un futur proche, une page non utilisée depuis un temps élevé a une probabilité faible d'être utilisée prochainement.
• L'algorithme choisit donc comme victime la page ayant fait l'objet de la référence la plus ancienne.
• La réalisation de l'algorithme impose d'ordonnancer les cases selon la date de dernière référence de la page qu'elles contiennent.

Algorithme NRU: *On prend ici en compte si la page a été référencée et modifiée!*

- Amélioration du LRU, en prenon cette fois en compte la modification des pages.
- On catégorise les pages suivant des classes:
    i. Non référencées, non modifiées.

    ii. Non référencées, modifiées.
    iii. Référencées, non modifiées.
    iv. Référencées, modifiées.
- Les pages de la classe 1 sont remplacées en priorité, puis on passe à la classe 2 etc..

Algorithme de la seconde chance:

- Modification de FIFO. Le but est d'éviter de supprimer des pages ayant des processus encore en cours.
- On va inspecter le bit R de chaque page, le bit R nous informe si la page a était utilisée récemment.
- Si le bit est à 0, on l'évince de la mémoire. Si la page a été modifiée, on l'écrit sur le disque, sinon on l'abandonne (on considère qu'il n'y a pas d'évolution de son état, du coup on ne la réecrit pas).
- Si le bit vaut 1, on place la page à la fin de la liste, et son temps de chargement est mis à jour avec le temps courant.

Algorithme de remplacement de l'horloge

- Un pointeur vers la page la plus ancienne.
- Quand un défaut de page se produit, la page pointée est testée. L'action entreprise dépend de la valeur du bit R.
    - Si R= 0, on retire la page
    - Si R=1, on met R à 0 et on avance le pointeur comme sur une horloge.

Notion d'écroulement:

- Un processus s'écroule lorsqu'il passe plus de temps à paginer qu'a exécuter
- Taux d'utilisation du processeur en fonction du degré de multiprogrammation (c-a-d en fonction du nombre de processus chargés en mémoire centrale) trop élevé => CHUTE.

Pagination sous Linux => voir diapo

Exercice => Pour FIFO on s'intéresse au premier de la date de chargement LRU => on s'intéresse à celui qui a été référencé en dernier, donc la plus petite valeur de Date Dern. Réf NRU => on s'intéresse à la plus petite valeur de rb