Ing1 **2002**
Systèmes d'Exploitation
TD 3

Les entrées-sorties

---

1 – Si une nouvelle technologie permettait d'avoir un temps d'accès au disque du même ordre de grandeur que le temps d'accès à la mémoire, quelles seraient les changements à apporter aux composants suivants du système d'exploitation pour profiter de ce gain temps d'accès ? Comparer ces changements avec les implémentations actuelles.

a - ordonnancement de processus
b - gestion de la mémoire
c - le pilote du disque pour la gestion des requêtes d'E/S

---

a – Maintenant le scheduler n'arrête plus l'exécution d'un processus même des E/S de courte durée. S'il s'agit de copier le disque entier sur une bande par exemple, l'UC est monpolisée longtemps pour un seul processus. Pour éviter cette situation, le scheduler doit pouvoir estimer la durée des E/S ou le SE doit permettre des commandes non bloquantes pour des opérations d'E/S de courte durée (écrire un bloc, lire un bloc).
Si l'on utilise le disque à la place de la mémoire centrale, on n'aura plus besoin d'algorithme d'ordonnancement global et on peut augmenter le degré de multiprogrammation. Dans ce schéma, on aura besoin d'un système de priorités pour assurer la priorité des E/S intéractives. Si les E/S deviennent aussi très rapides, on aura besoin de modifier l'algorithme d'ordonnancement à priorités

b – 3 stratégies :

b1 – éliminer complètement la mémoire et utiliser le disque. En stockant les processus en contigu ou dans des segments contigus et en limitant la taille d'un processus à la taille du disque, on peut éliminer le recours aux adresses virtuelles (pour les accès contigus) ou le réduire à un segment et un offset (pour les segments contigus). La protection mémoire des processus peut être implémentée par le hardware (registres base et taille). Le temps d'accès devient égal au temps d'accès du disque. D'autre part, on n'a plus besoin de pagination car la mémoire disque est supposée très grande.

b2 – Si l'on veut une mémoire virtuelle plus grande que le disque, on aura besoin de la hiérarchie mémoire et des algorithmes de gestion mémoire comme la pagination.
Si l'on considère un système à pagination avec la table des pages en mémoire et le même temps d'accès pour la mémoire et le disque, le temps d'accès sera (2 accès mémoires pour un *hi*t – un à la table des pages et un à la mémoire ; et 4 accès mémoire pour un *miss* – un à la table des pages, un pour écrire une page modifiée sur le disque, un pour charger la page dans la mémoire et un pour accéder à la page) :
temps d'accès : 2*ma + 4*ma (temps pour le premier bloc nécessaire et le reste de la page est traitée en background)
En ajoutant un TLB, le temps d'accès devrait diminuer et devient :
temps d'accès = p*ma + (p+f – 1)*2*ma + f*4*ma où
p : probabilité que la page soit dans le TLB
f : probabilité que la page soit sur le disque (pas dans la mémoire)
En utilisant le disque, le temps d'accès est encore au moins 4 fois plus élevé.
On veut alors limiter le taux de défaut de pages et on peut continuer à utiliser *clock algorithm*. Toutefois, il faudrait mesurer le taux de défaut de pages avec des algorithmes simples et rapides comme FIFO et comparer les performances.

b3 – Si l'on veut utiliser la mémoire et le disque en même temps, il faut trouver le moyen de faire la translation d'adresses.

c – Le pilote n'a pas besoin d'intelligence, comme l'optimisation des recherches, parce que les pistes et les secteurs ne sont plus nécessaires. Comme maintenant un processus peut continuer à s'exécuter même s'il accède au disque, le pilote devient un chemin critique et il faut qu'il exécute le minium de traitement possible, en traitant par exemple les E/S en FIFO. Si l'on élimine complètement l'utilisation de la mémoire, il serait judicieux de faire migrer le pilote vers l'UC.

2 – On considère un ordinateur avec une UC et un périphérique D d'E/S connecté à la mémoire centrale M via un bus partagé et un bus de données d'un mot. l'UC peut exécuter au maximum 10^6 instructions par seconde. Une instruction requiert 5 cycles machine en moyenne dont 3 utilisent le bus mémoire. Une opération mémoire READ ou WRITE utilise un cycle machine. On suppose que l'UC exécute en continue des programmes qui monopolisent 95% de son temps d'exécution sans instructions d'E/S.
On veut transférer de grands blocs de données entre M et D :

a- si l'on utilise les E/S programmées (*programmed I/O*) et que le transfert d'un mot requiert 2 instructions, estimer le taux de transfert maximal R_{max} entre M et D
b- estimer R_{max} si le transfert par DMA est utilisé

ooooooooooooooooooo

a – Avec les E/S programmées, on ne peut utiliser les E/S que lorsque l'UC n'est pas occupée parce qu'elle est impliquée dans le transfert. Comme elle est occupée à 95%, on ne peut transférer les données entre M et D que 5% du temps :
0.05 * (10^6 instr/s) = 50000 instr/s
Comme le transfert d'un mot demande 2 instructions, on peut alors transférer au maximum :
Rmax = 50000 * 0.5 = 25000 mots/s
b – Avec DMA, on peut utiliser le bus quand l'UC est inoccupée pendant les 5 cycles et on peut aussi utiliser le bus quand l'UC est occupée pendant 2 cycles sur 5 :
0.5 * (10^6 instr/s) * (5 cycles/instr) + 0.95 * (10^6 instr/s) * (2 cycles/5) =
2.15 * 10^6 cycles/s
Un mot peut être transféré en 1 cycle, donc on aura un taux de transfert max :
Rmax = (2.15 * 10^6 cycles/s) * (1 mot/cycle) = 2.15 * 10^6 mots/s

3 – Un ordinateur dispose d'un cache de C mots, d'une mémoire centrale de M mots et d'un disque utilisé pour la mémoire virtuelle. Si un mot est dans le cache, son temps d'accès est t_c ns. S'il est dans la mémoire mais pas dans le cache, t_m ns sont nécessaires pour le charger dans le cache suivi de t_c pour charger un registre de l'UC à partir du cache. Si le mot n'est pas dans la mémoire, t_d ns sont nécessaires pour le charger à partir du disque suivi de t_m et t_c. Si la probabilité de trouver un mot dans le cache (*cache hit ratio*) est (C – 1)/C et la probabilité de le trouver dans la mémoire est (M – 1)/M, calculer le temps d'accès moyen d'un mot?

°°°°°°°°°°°°°°°°°°°°°°°°°°°°

Soit $P_c = (C - 1) / C$ et $P_m = (M - 1)/M$
le temps d'accès moyen T_{am} est :

$$T_{am} = P_c * t_c + (P_m * (t_m + t_c) + (1 - (P_c + P_m)) * (t_d + t_m + t_c)$$
$$= t_c + t_m (1 - P_c) + t_d (1 - P_c - P_m)$$
$$= t_c + t_m (1/C) + t_d (1/C - (M - 1)/M)$$

4 – Soit un disque composé de 16 faces (8 plateaux double face), 110 pistes/face, 64 secteurs/piste, 1024 octets/secteur. Le disque a une vitesse de rotation de 300 tours/mn et le contrôleur du disque dispose d'un buffer de 1 Ko. L'écriture et la lecture du buffer sont exclusives.
L'horloge a une fréquence de 50 MHz, une instruction s'exéxute en 5 cycles d'horloge en moyenne et l'accès au bus est nécessaire avec 3 cycles/instruction en moyenne
a - quelle est la capacité du disque en Ko ?
b - il faut 100 instructions (setup) et 2 cycles/octet (transfert mémoire/buffer) pour traiter l'interruption disque. En négligeant les délais de transfert mémoire, calculer le taux de transfert mémoire/disque maximal dans le cas des E/S gérées par interruption (*interrupt-driven I/O*).

ooooooooooooooooooooo

a – Capacité du disque : 16*110 * 64 * 1Ko = 112640 Ko
b – Le disque est équivqlent à un seul plateau avec 16*1024 octets/secteur. le buffer contient le 1/16 de cette valeur et peut être rempli en (1/16)*(1/64)*(60/360) = 163 µsec. Une fois le buffer rempli, le contrôleur du disque émet une interruption pour transférer les données. Cela dure environ 5*100 cycles pour le setup et 2*1024 cycles pour transférer tout le buffer, ce qui fait 2548 cycles, environ 51 µs. Le temps total (disque + transfert) pour transférer 1 Ko est environ 214 µs. Le meilleur taux de transfert moyen est donc de 4.56 Mo/s.

5 – Un système utilise la politique de la page à la demande (*demand paging*) pour l'allocation mémoire. Sous une certaine charge, on a relevé que l'UC est utilisée 20% du temps et que le disque de pagination est utilisé 99% du temps.
Parmi les éléments suivants, quels sont ceux qui peuvent améliorer l'utilisation de l'UC ? Expliquer pourquoi.
a - un processeur plus rapide
b - un disque avec une capacité plus grande
c - un disque plus rapide
d - augmentation du degré de multiprogrammation
e - diminution du degré de multiprogrammation

ooooooooooooo

a - Une UC plus rapide va encore diminuer l'utilisation de l'UC car le système est plutôt limité par la vitesse du disque de pagination.
b – La grande taille du disque permet un espace virtuel plus grand ou permet d'avoir plus de processus à s'exécuter, mais ne permet pas d'augmenter l'utilisation de l'UC. Si le temps de recherche est augmenté, c'est plutôt une diminution qui se produirait.
c – Un disque plus rapide permet d'accélerer le traitement des requêtes de pagination et peut probablement augmenter ll'utilisation de l'UC
d – Augmenter le degré de multiprogrammation peut augmenter l'utilisation de l'UC si les tâches ne font pas beaucoup de pagination. Vraisemblablement, cela va allonger la queue d'attente du disque et ralentir le système (diminuer l'utilisation de l'UC)

e – Diminuer le degré de multiprogrammation devrait augmenter l'utilisation de l'UC si le système était en état d'écroulement (*thrashing*) parce que trop de tâches étaient en compétition pour l'obtention des pages mémoire.

6 - Citer 3 types de cache et leur implémentation (hardware, SE ou programme utilisateur).

ooooooooooooo

7 - Des requêtes de cylindres arrivent dans le pilote du disque dans l'ordre suivant : 10, 22, 20, 2, 40, 6, 38. La tête du disque est initialement positionnée sur le cylindre 15. Si le temps de recherche (*seek time*) est de 6 ms par cylindre, calculer le temps de recherche nécessaire avec l'algorithme SSTF ?