Algorithmes d'ordonnancement de tâches et de gestion de ressources

Motivation 1 Le système d'exploitation offre une abstraction du matériel. Il doit donc gérer les ressources qui y sont associées (temps de calcul, mémoire vive, mémoire persistante…).

I. Gestion du temps processeur [MOS 2.5] [OSC 5]

Définition 2 **Un processus** est une instance d'exécution d'un programme, incluant son espace mémoire et l'état des registres.

Définition 3 On dit d'un processus qu'il est **endormi** s'il est en attente d'une entrée-sortie (lecture/écriture sur mémoire persistante, affichage, touche clavier ou clic souris, …) ou s'il a demandé à être endormi. Un processus qui ne dort pas est **prêt**.

Définition 4 **Un cœur de calcul** est la partie du processeur capable d'exécuter un processus (registres, unité de traitement, …).

Problème 5 **Ordonnancement** S'il y a plus de processus prêts que de cœurs de calcul, il faut choisir **quel processus exécuter**. C'est le rôle de l'**ordonnanceur**, qui fait partie du système d'exploitation.

A. Objectifs et hypothèses sur l'ordonnanceur [MOS 2.5]

Définitions 6 **Objectif commun : équité** Tous les ordonnanceurs visent à garantir une répartition équitable du temps processeur.

Définition 7 **La préemption** est le fait d'interrompre un processus en cours d'exécution. Elle est un outil pour garantir l'équité.

Définition 8 **Lors d'un changement de contexte** on sauvegarde l'état du processus en cours d'exécution, et on charge l'état d'un autre processus à exécuter.

Cas particulier 9 **Dans le calcul par lots,** s'ajoutent les objectifs

- de maximiser le nombre de travaux complétés par heure,
- de minimiser le temps entre soumission et complétion,
- et d'utiliser au maximum les unités de calcul disponibles.

Dans ce contexte, on connaît également à l'avance le temps maximal que chaque processus prendra à s'exécuter.

Cas particulier 10 **Dans les contextes interactifs** s'ajoute l'objectif de minimiser le temps entre la création d'un processus et sa première exécution (on parle de **réactivité**). De plus, l'utilisateur·ice peut assigner différentes priorités aux processus.

B. Ordonnancement naïf

Définition 11 **Premier Entré, premier Sorti (PEPS)** est la politique d'ordonnancement la plus simple: les processus sont exécutés dans leur ordre de création, sans préemption.

Éxemple 12 Supposons avoir un seul cœur de calcul, et deux processus: A met 81 minutes à s'exécuter et est arrivé en premier, B met 19 minutes à s'exécuter et est arrivé 1 minute après A.

- La politique PEPS n'est pas satisfaisante pour le calcul par lot, puisque le temps de complétion moyen sera de 50 minutes, là où interrompre A pour exécuter B donne un temps de complétion moyen de 35 minutes.
- La politique PEPS n'est pas satisfaisante pour un contexte interactif non-plus, puisque B devra attendre 80 minutes avant sa première exécution.

C. Ordonnancement pour le calcul par lot

Défintion 13 **« Plus court d'abord »** est une politique d'ordonnancement non-préemptive exécutant d'abord les processus les plus courts.

Définition 14 **« Plus courte complétion d'abord »** est la version préemptive de la politique précédente : quand une nouvelle tâche arrive, si elle est plus courte que le temps restant à la tâche courante, cette dernière est interrompue.

D. Ordonnancement dans les contextes intéractifs

Définition 15 **La politique « Chacun son tour »** aussi appelée « tourniquet » divise le temps processeur en segments de même taille, et les attribue aux différents processus de façon cyclique.

DEV

Définition 16 [OSTEP 8] **« Files de rétroaction multi-niveau »** est une politique qui s'adapte au comportement des processus.

II. Gestion des pages mémoire [MOS 4.3 à 4.8] [OSC 9.3 et 10]

Définition 17 Pour protéger les processus l'un de l'autre, et pour simplifier leur utilisation de la mémoire, le système d'exploitation **virtualise la mémoire**. Chaque processus a l'impression d'avoir accès à une grande mémoire monolithique.

A. Principes de la pagination

Définition 18 Les mémoires (virtuelle ou physique) sont découpées en blocs de taille fixe appelés des **pages**.

Définition 19 Pour chaque processus, le système d'exploitation dispose d'une **table des pages** donnant la correspondance entre pages virtuelles et pages physiques.

Définition 20 **L'unité de gestion de la mémoire** (MMU en anglais) est le composant physique qui, à chaque accès mémoire, traduit les adresses virtuelles en adresses physiques, en utilisant les informations de la table des pages. Il dispose d'un cache, appelé **Table Lookaside Buffer** (TLB).

Définition 21 Pour accélérer le chargement initial d'un programme et pour économiser de la place en mémoire RAM, on peut faire de **la pagination à la demande**: ne charger une page que lorsqu'elle est utilisée.

Problème 22 Quand il n'y a plus de place libre en RAM pour charger une page, il faut décider quelle page évincer. Ce problème s'appelle le **remplacement de pages**.

B. Remplacement de pages

Définition 23 **Le taux d'erreur** d'une politique de remplacement de page est la proportion de fois où une page utilisée n'était pas déjà en RAM, pour une séquence donnée d'accès mémoire.

Définition 24 Pour une séquence d'accès mémoire donnée, **la politique « optimale »** évince la page qui sera ré-utilisée le plus tard possible dans la séquence d'accès (ou qui ne sera pas ré-utilisée).

Remarque 25 La politique « Optimale » n'est pas réalisable en pratique, car le système d'exploitation ne connaît pas le futur, et donc ne sait pas quelle page sera réutilisée le plus tard possible dans par le processus. Par contre, « Optimale » sert d'outil théorique de comparaison, pour évaluer l'efficacité des politiques de remplacement de page.

DEV

Définition 26 **La politique « Première Entrée, Première Sortie »** consiste à maintenir une file des pages. Lorsqu'on doit évincer une page, on choisit celle qui a été chargée en mémoire il y a le plus longtemps.

Remarque 27 Avec la politique « Première Entrée, première sortie », le fait de rajouter des ressources (taille de la mémoire RAM) peut dégrader les performances (taux d'erreurs de page). Ce phénomène est appelé **anomalie de Belady**.

Définition 28 **La politique « Moins Récemment Utilisée »** évince la page la moins récemment utilisée. Elle repose sur l'heuristique qu'une page récemment utilisée a plus de chances d'être à nouveau utilisée.

Remarque 29 **Les politiques de remplacement de page doivent s'exécuter très rapidement** car le temps que ces politiques s'exécutent, les programmes de l'ordinateur ne s'exécutent pas.

Définition 30 **L'algorithme de l'Horloge,** ou « algorithme de la deuxième chance » est une politique de remplacement de page qui permet d'approcher la politique « Moins Récemment Utilisée » mais qui s'exécute plus rapidement. Le principe est de mettre les pages dans une liste circulaire et de les marquées comme récemment utilisées ou pas. La page évincée sera la plus anciennement chargée, sauf si elle a été utilisée depuis la dernière fois où on a essayé de l'évincer.

III. Ordonnancement des accès à un disque dur [OSTEP 37.5]

Problème 31 Le temps de réponse d'un disque dur est essentiellement dû à des contraintes physiques (déplacement de la tête de lecture, rotation du disque). Il est donc important d'organiser les requêtes pour minimiser ce temps de réponse. Les requêtes sont alors stockées dans une mémoire tampon en attendant d'être traitées.

Algorithme 32 Shortest Seek Time First (SSTF) Les requêtes sont organisées de sorte à minimiser les déplacements de la tête de lecture en favorisant les accès sur la même piste.

Remarque 33 Famine SSTF peut faire apparaître une situation de **famine** : certaines requêtes ne seront jamais satisfaites.

DEV

Algorithme 34 SCAN Balayage du disque dans un sens, puis dans l'autre, en traitant les requêtes lorsque la tête de lecture passe sur la zone concernée.

Remarque 35 Cet algorithme est moins efficace, mais résouds le problème de famine. Cependant, il a tendance à favoriser les blocs mémoires présents sur les pistes au centre.

Algorithme 36 C-SCAN Variante de SCAN qui parcourt le disque toujours dans le même sens. Cela permet un traitement plus uniforme des requêtes

Remarque 37 Aujourd'hui, les disques à mémoire flash permettent de se contenter de variantes simples de l'algorithme PAPS, en évitant les contraintes mécaniques des disques durs.

IV. Gestion de la mémoire par les programmes

Problème 38 Les programmes se voient accordés un espace d'adressage qu'ils peuvent utiliser librement. Dans un soucis d'efficacité, il est important de réduire l'empreinte mémoire d'un programme sur le système en libérant ou réutilisant les allocations qui ne sont plus utiles.

A. Gestion manuelle [OSTEP 14]

Méthode 39 Gestion manuelle Gestion de la mémoire au moyen d'une API système (`malloc`, `free`, …).

Problème 40 Bin packing L'allocateur mémoire peut utiliser différentes stratégies pour choisir où allouer la mémoire, comme *First-Fit* (premier emplacement), ou *Best-Fit* (meilleur emplacement).

Remarque 41 Bien qu'efficace, la gestion manuelle de la mémoire peut être source d'erreurs pouvant rendre le programme **vulnérable à des failles de sécurité**.

B. Gestion automatique [GC]

Méthode 42 Gestion automatique On alloue la mémoire nécessaire au stockage d'une donnée à sa création, puis on choisi quand la libérer **algorithmiquement**. Cet algorithme est appelé **ramasse-miette**.

Algorithme 43 Comptage de références On peut ajouter à chaque allocation mémoire un compteur permettant de savoir combien d'autres données y font référence. Quand ce compteur atteint 0, on libère la mémoire.

Remarque 44 Bien que simple à implémenter, le comptage de références à de nombreux désavantages, comme la non-détection des cycles, la quantité potentielle de mises à jour du compteur, ou son interaction avec des applications concurrentes.

DEV

[GC 2]

Méthode 45 Traçage Garder en mémoire un *graphe* de références que l'on peut traverser depuis sa source pour savoir quelles données sont *accessibles*.

Algorithme 46 Mark-And-Sweep pars des données *racines* (variables locales, statiques, globales) puis traverse le graphe de références pour marquer les données accessibles. Les données non-marquées sont ensuite libérées.

Exemple 47 Le module `Gc` en OCaml permet d'accéder aux statistiques de son ramasse-miette.

Algorithmes d'ordonnancement de tâches et de gestion de ressources

1 Motiv

I. Gestion du temps processeur [MOS 2.5] [OSC 5]

2 Def Un processus
3 Def Processus endormi
4 Def Un cœur de calcul
5 Prob Ordonnancement

A. Objectifs et hypothèses sur l'ordonnanceur [MOS 2.5]

6 Définitions Objectif commun : équité
7 Def La préemption
8 Def Lors d'un changement de contexte
9 Cas particulier Dans le calcul par lots,
10 Cas particulier Dans les contextes interactifs

B. Ordonnancement naïf

11 Def Premier Entré, premier Sorti (PEPS)
12 Exemple

C. Ordonnancement pour le calcul par lot

13 Défintion « Plus court d'abord »
14 Def « Plus courte complétion d'abord »

D. Ordonnancement dans les contextes intéractifs

15 Def La politique « Chacun son tour »
16 Def [OSTEP 8] « Files de rétroaction multi-niveau »

II. Gestion des pages mémoire [MOS 4.3 à 4.8] [OSC 9.3 et 10]

17 Def Mémoire virtuelle

A. Principes de la pagination

18 Def Pages
19 Def Table des pages
20 Def L'unité de gestion de la mémoire
21 Def Pagination à la demande
22 Prob Remplacement de page

B. Remplacement de pages

23 Def Le taux d'erreur
24 Def La politique « Optimale »
25 Rem « Optimale » est outil théorique
26 Def La politique « Première Entrée, Première Sortie »
27 Rem Anomalie de Belady
28 Def La politique « Moins Récemment Utilisée »
29 Rem Les politiques de remplacement de page doivent s'exécuter très rapidement
30 Def L'algorithme de l'Horloge,

III. Ordonnancement des accès à un disque dur [OSTEP 37.5]

31 Prob
32 Algo Shortest Seek Time First (SSTF)
33 Rem Famine
34 Algo SCAN
35 Rem
36 Algo C-SCAN
37 Rem

IV. Gestion de la mémoire par les programmes

38 Prob

A. Gestion manuelle [OSTEP 14]

39 Métho Gestion manuelle
40 Prob Bin packing
41 Rem

B. Gestion automatique [GC]

42 Métho Gestion automatique
43 Algo Comptage de références
44 Rem
45 Métho Traçage
46 Algo Mark-And-Sweep
47 Ex Le module `Gc`

Bibliographie

[MOS] Tanenbaum & Bos, *Modern Operating System, 5th Edition.*

[OSC] A. Silberschatz & P. B. Galvin & G. Gagne, *Silberschatz's Operating System Concepts, Global Edition.*

[OSTEP] R. H. Arpaci-Dusseau & A. C. Arpaci-Dusseau, *Operating Systems, three easy pieces.*

[GC] R. Jones & A. Hoskings & E. Moss, *The Garbage Collection Hand Book: the art of automating memory management.*

Notes GC:

- Le livre en référence est une mine d'or de devs (peut-être pas *tous* au programme).
- Se replace en leçon graphe
- Pour la deuxième partie (tri-marquage) du dev, peut-être remplacée par :
  - Mark-And-Sweep concurrent (parties 15/16) pour placer en synchro
  - Stop-And-Copy (partie 4)

Santiago Sara Bautista & Aloïs Rautureau