620 Mémoire virtuelle

INF3173
Principes des systèmes d'exploitation

Jean Privat

Université du Québec à Montréal

Hiver 2021

Mémoire virtuelle

Aller plus loin ?

- Offrir à chaque processus une mémoire plus grande que celle disponible
- Utiliser le disque comme mémoire supplémentaire
- Un processus n'a pas forcément besoin d'être entièrement en mémoire principale

$\rightarrow$ De façon transparente pour les processus

Partitions et fichiers d'échanges

- Alias: *swap*
- Fichier ou partition dédiés
- Utilisée comme mémoire supplémentaire
- Accès **lent**, donc à utiliser correctement

Mémoire virtuelle sans pagination (historique)

- Alias: *swapping* de processus

Quand la mémoire est faible

- Trouver un processus pas souvent actif
- Copier toute sa mémoire sur disque (*swap out*)
- Puis libérer la mémoire du processus

$\rightarrow$ La mémoire n'est plus faible !

Quand on doit continuer l'exécution du processus

- On recharge le processus en mémoire (*swap in*)
- Quitte à *swap out* un autre processus pour faire de la place

Mémoire virtuelle et pagination

Paging (*swapping* de page)

- Une page virtuelle peut être
  - Soit en mémoire physique
  - Soit sur le disque (en *swap*)
  - Soit invalide
- Si RAM est pleine: on sauve
  - On **descend** des pages physiques vers le disque (*page out*)
- Si accès à une page virtuelle qui est sur le disque: on charge
  - On **monte** une page physique depuis le disque (*page in*)

Avantages

- Granularité beaucoup plus fine que le *swapping* de processus
- Chargement et déchargement de morceaux de processus au besoin

$\rightarrow$ On y reviendra (fonctionnalités avancées)...

Mémoire résidente vs. mémoire virtuelle

Mémoire virtuelle

- Les pages virtuelles de l'espace mémoire utilisable d'un processus
- Code + données + pile + tas + bibliothèques + etc.

$\rightarrow$ Ce qui apparait avec `pmap(1)`

Mémoire résidente

- Les pages d'un processus physiquement en RAM
- Transparent pour les processus, géré par le noyau
- Habituellement, une page physique est comptée une fois même si associée à plusieurs pages logiques

Question

Qu'est-ce qui peut être plus grand que la taille de la RAM ?

- La taille de la mémoire virtuelle d'un processus
- La taille de la mémoire résidente d'un processus
- La somme des tailles de la mémoire résidente des processus

Mise en œuvre

Côté MMU : on ne change rien

- Pas besoin de changer de processeur
- Tout se fait côté système d'exploitation

Pagination cotée MMU (rappel)

- La table des pages (MMU) indique seulement
  - Si une page logique existe
  - Et si oui : où (quelle page physique) et avec quels droits
- Une faute CPU est lancée
  - Si le CPU accède à une page logique absente
  - Si le CPU accède à une page logique avec les mauvais droits

Côté système d'exploitation

Migration d'une page sur disque

- Quand le système d'exploitation migre une page
- Il marque que la page est en swap (et où)
  - Ça ne rentre pas dans la table des pages
  - Le système a ses propres structures de données
- Il met à jour la table des pages pour invalider la page logique

$\rightarrow$ Coût: copie sur disque et mise à jour de la table des pages

Côté système d'exploitation

Accès du processus à une page virtuelle en RAM

- MMU traduit correctement adresse logique en adresse physique
- Le CPU travaille normalement (rien de spécial)

$\rightarrow$ Surcoût: 0

Accès du processus à une page virtuelle invalide

- MMU lève une faute CPU (faute de page)
- Le système d'exploitation
  - Attrape l'interruption matérielle
  - Détermine que la page virtuelle est invalide
  - Envoie SIGSEV au processus
- Le processus est terminé (ou gère le signal)

$\rightarrow$ Surcoût: une vérification en plus

Accès du processus à une page virtuelle en swap

- MMU lève une faute CPU (faute de page)

Le système d'exploitation

- Attrape l'interruption matérielle
- Détermine que la page virtuelle est en fait en swap
- Lance le chargement dans une page physique
- (et éventuellement la migration d'une autre page si pas de place...)
- Passe le processus à bloqué (et appelle l'ordonnanceur)

Quand le chargement est fini, le système d'exploitation

- Met à jour la table des pages
- Passe le processus à prêt (et appelle l'ordonnanceur)

Lorsqu'élu par l'ordonnanceur, le processus

- Recommence l'instruction fautive
- Qui réussit (cette fois)

Défaut de page

Défaut **majeur** de page

- L'adresse virtuelle est valide
- Mais la page n'est pas en mémoire : elle est sur disque
- Il faut faire des entrées-sorties pour la récupérer
- Métrique `%F` de `time(1)`

→ le système charge la page en mémoire (couteux)

Défaut **mineur** de page

- L'adresse virtuelle est valide
- Or page physique est en mémoire (cache ou chance)
- Mais n'est pas associée dans la table des pages
- Métrique `%R` (*recoverable*) de `time(1)`

→ le système met juste à jour la table des pages (peu couteux)

Algorithmes de remplacement

Problème bien étudié et bien généralisable (swap, cache, etc.)

Données

- Un grand nombre de pages virtuelles
- Une séquence de demandes de pages virtuelles
- Un nombre limité de pages physiques

Objectif

- Trouver à chaque demande quelle page physique utiliser
- Déterminer quelle page migrer quand la mémoire est pleine
- Minimiser le nombre de défauts de pages (et de migration)

Idées de base : quelles pages migrer ?

- Idéal : Les pages non utilisées dans un futur proche
- Approximation : Les pages non utilisées récemment
- Approximation pire : Les pages anciennement allouées

Algo naïf : file d'attente (FIFO)

Principe

- Les pages vieilles migrent en swap

Exercice

- Séquence: 1,2,3,4,1,2,5,1,2,3,4,5
- Avec 3, puis 4, pages physiques

Algo naïf : file d'attente (FIFO)

Principe

- Les pages vieilles migrent en swap

Exercice

- Séquence: 1,2,3,4,1,2,5,1,2,3,4,5
- Avec 3, puis 4, pages physiques

Anomalie de Belady (curiosité théorique)

- Avec plus de pages physiques (plus de RAM)
- Le nombre de fautes ne décroit pas forcément
- Dans certaines situations, le nombre de fautes peut augmenter

Algo de l'horloge (ou de la seconde chance)

Comme FIFO, mais un bit indique s'il y a eu utilisation

- Un bit marque les pages utilisées (MMU)
- Parcours circulaire des pages candidates
- Si son bit $= 1$, on le passe à 0 sinon on migre la page en swap

Exercice

- 1,2,3,4,1,2,5,1,2,3,4,5 avec 3 et 4 pages physiques