Travail pratique #3 - IFT-2245

Nezha Chahid et Dejla Ben Ltaief P1069712

April 24, 2016

1 Introduction

Dans ce travail pratique, nous devons implémenter en langage C un programme qui simule un gestionnaire de mémoire virtuelle par pagination (paging). En simulant des accés mémoire consécutifs en traduisant les adresses logiques en adresses physiques de 16 bits.

2 Implémentation

2.1 Lecture

Nous avons commencée par trouver la page et l'offsset d'une adresse logique donnée. On établit un lookup, si la page se trouve alors le tlb hit le found count sont est incrementés. Si la page n'est pas présente dans le tlb , alors on cheche dans la table des pages . Si on la trouve page found count et miss count sont incrementes. Si nous avons un miss et un page fault alors, on fait appel a la fonction demand-page. Finallement si le TLB est rempli, alors nous utilisons un algorithme de remplacement.

2.2 Ecriture

Clairement, c'est la meme implementation qu'on as fait pour la lecture sauf quelques petits changements, Nous ajoutons la sauvegarde du caractére a la mémoire.

3 réponses aux questions

3.1 Question1

Pour le tlb nous avons utilisée deux algorithmes,le premier FIFO, et le second LFU.

1. Algorithme FIFO
   Nous avons choisi l'algorithme de remplacement FIFO parce qu'il est trés simple a implémenter. un attribut age a été ajouté a la structure, a chaque fois qu'une entrée tlb est ajouté au tlb, ageCounter est incrementée. si un remplacement doit être fait, on choisit tout simplement celle avec l'age minimale.

   Avantages du FIFO: Rapidité. pas de probléme de famine.

   Inconvènients du FIFO:
   il ne prend pas en compte lutilitè des pages,il prend pas en compte les pages qui sont rècemmnent utilisèes, ni les pages qui sont frèquemment utilisèes, une page qui est utilisèe une seule fois peut remplacer une page qui est frèquement utilisèe.
   Anomalie de Belady : on peut dans certains cas augmenter les èchanges necèssaires si on augmente la taille de la mèmoire.

2. Algorithme LFU
   la moins souvent utilisèe : on garde un compteur qui est incrementèe à chaque fois que le cadre est referencèe, et la victime sera le cadre dont le compteur est le plus bas. Nous l'avons implementèe de la facon suivante: A chaque fois qu'une entrèe est ajoute au tlb pour la première fois, L'attribut use est a 1. A chaque fois que la variable use est incrementèe. si l'entrée sort du TLB table et elle s'introduit après ,la variable use va etre remise a 1.
   Avantages du LFU:
   Facilité d'implémentation.
   Inconvènients du LFU:
   Cet algorithme prèsente un problème quand une page est très utilisé pendant la phase initiale d?un programme, mais qu?elle n?est plus employèe à nouveau par la suite. Comme elle a ètè amplement utilisée, elle posséde un grand compte et reste donc en mèmoire m ême si elle n?est pas utilisèe.

3.2 Question2

Nous avons effectuer quelques modifications au fichier command.in, et nous avons effectuée les tests suivants.

1. Fifo applique

```
tlb hits:   9
tlb miss:   161
tlb hit ratio:   0.052941
page found: 46
page fault: 124
page fault ratio:   0.729412
```

2. LFU applique

```
tlb hits:   12
tlb miss:   158
tlb hit ratio:   0.070588
page found: 46
page fault: 124
page fault ratio:   0.729412
```

3. Fifo applique

```
tlb hits:   762
tlb miss:   86
tlb hit ratio:   0.898585
page found: 776
page fault: 72
page fault ratio:   0.084906
```

4. LFU applique

```
tlb hits:   765
tlb miss:   83
tlb hit ratio:   0.902123
page found: 776
page fault: 72
page fault ratio:   0.084906
```

La pageFault ratio est la m ême pour les deux algorithmes, ceci est pr èvu, comme autrefois la page est chargèe en mèmoire, et que la mèmoire est suffisamment grande pour stocker les donnèes, alors aucun èchange doit être effectu èe. Par ailleurs nous remarquons une diffèrnce entre tlb hit ratio pour LFU est FIFO dans les deux cas.LFU bat FIFO de 2 pourcent dans le premier test 1 pourcent dans le second test. Finalement , LFU est plus efficace que FIFO.

3.3 Question3

resultats-FIFO
Sans les changements

```
tlb hits:    6
tlb miss:    94
tlb hit ratio:    0.060000
page found: 21
page fault: 79
page fault ratio:    0.790000
```

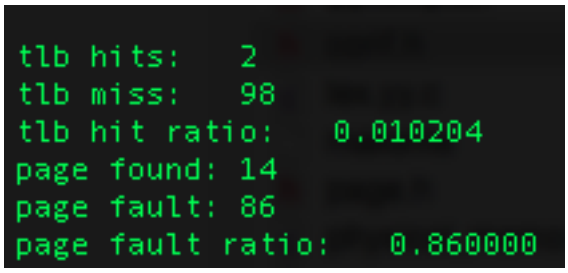
Avec les changements

```
tlb hits:    1
tlb miss:    99
tlb hit ratio:    0.010000
page found: 21
page fault: 79
page fault ratio:    0.790000
```

Configuration

```
#define NUM_FRAMES 256*2
#define NUM_PAGES 256
#define PAGE_FRAME_SIZE 128
#define TLB_NUM_ENTRIES 8
#define PHYSICAL_MEMORY_SIZE (NUM_FRAMES * PAGE_FRAME_SIZE)
```

Nous avons fait des tests sur 100 elements D'après les images ci dessous, nous remarquons que pour l'algorithme fifo que la situation se degrade en modifiant le tlb entries a 8 le tlb-miss est passee de 96 a 99.
2 éme test:
Nous avons modifié le num frames a 512 et le ratio du Fifo a diminué , ce qui confirme son incovenient cité ci dessus ce qui n'est pas le cas pour LFU ou le hitratio reste le meme.
Avec les changements



Nous avons essayé de modifié le NUMPAGES en l'augmentant mais nous n'avons pas eu des changements dans les deux cas.

3.4 References

http://stic-os.webs.com/Support_Cours/Algorithmes%20de%20remplacement.pdf
Un ancien Tp
Plusieurs notes de cours