hw4-Memory-Manager-template

> 成大資訊三乙 F74082264 陳彥博

Different Policy

1. `TLB(Translation Lookaside Buffer)` Replacement Policy
   - `RANDOM`:
   - `LRU`: Least Recently Used
2. `PTB` Replacement Policy
   - `FIFO`: First-In First-Out
   - `CLOCK`: Second Chance
3. `Physical Frame` Allocation Policy
   - `GLOBAL`:
     - Each Process shares a clock
   - `LOCAL` :
     - Each Process has its clock (Individual)

Analysis (Simulate at Sample Code)

1. LRU, FIFO, GLOBAL

- Result

```
Process A, Effective Access Time = 164.758
Process A, Page Fault Rate = 0.723
Process B, Effective Access Time = 163.709
Process B, Page Fault Rate = 0.665
```

2. LRU, FIFO, LOCAL

- Result

```
Process A, Effective Access Time = 164.980
Process A, Page Fault Rate = 0.774
Process B, Effective Access Time = 163.144
Process B, Page Fault Rate = 0.700
```

3. LRU, CLOCK, GLOBAL

- Result

```
Process A, Effective Access Time = 164.758
Process A, Page Fault Rate = 0.723
Process B, Effective Access Time = 163.709
Process B, Page Fault Rate = 0.665
```

4. LRU, CLOCK, LOCAL

- Result

```
Process A, Effective Access Time = 164.980
Process A, Page Fault Rate = 0.774
Process B, Effective Access Time = 163.522
Process B, Page Fault Rate = 0.694
```

5. RANDOM, FIFO, GLOBAL

  - Result

```
Process A, Effective Access Time = 164.309
Process A, Page Fault Rate = 0.723
Process B, Effective Access Time = 162.953
Process B, Page Fault Rate = 0.665
```

6. RANDOM, FIFO, LOCAL

  - Result

```
Process A, Effective Access Time = 164.980
Process A, Page Fault Rate = 0.774
Process B, Effective Access Time = 162.761
Process B, Page Fault Rate = 0.700
```

7. RANDOM, CLOCK, GLOBAL

  - Result

```
Process A, Effective Access Time = 165.200
Process A, Page Fault Rate = 0.723
Process B, Effective Access Time = 163.177
Process B, Page Fault Rate = 0.665
```

8. RANDOM, CLOCK, LOCAL

  - Result

```
Process A, Effective Access Time = 164.980
Process A, Page Fault Rate = 0.774
Process B, Effective Access Time = 162.956
Process B, Page Fault Rate = 0.694
```

Conclusion

> Evidently, the Effective Access Time (EAT) is no mush different in these status.

EAT and Page Fault Rate

影響 EAT (Effective Access Time) 的原因

> TLB Policy 會影響 EAT

1. 時間增加
   - Process 的交換次數上升
     - 因為每次交換 Process 時 TLB 會被清空，因此 TLB Miss 的機會增加。
   - Process 的數量增加
     - 因為越多的process，清空的機會變高
   - OS 將 CPU 均勻的分配給不同process
     - 交換率提升
2. 減少時間
   - Process 數量變少
     - 清空機率變低，進而減少交換次數
   - OS 在每一段時間集中處理某一個 process 的東西
     - 減少交換次數

影響 Page Fault Rate的原因

> Page Replacement Policy 和 Frame Allocation Policy

1. 降低 rate
   - Process數增加
     - 使用的 Virtual Memory 增加，因此被 Swap 到 Disk 的機會變高。
2. 增加 rate
   - Process 使用到的memory 比較少
     - 當 Physical Memory 的 Frame 就能夠滿足所有的 Process ，那 rate 也會降低
   - Process 使用到的 Virtual Memory (Page) 比較少
     - 重複取用相同的 Page(frame)，會減少 page Fault 的次數

Different Policy

1. TLB Replacement Policy：(**影響EAT**)
   - 我認為，TLB在 `LRU` 和 `Random` 間的差異不大的原因可能是因為測資的偏差性
     - 會增加兩者的差異性：
       - 有 極為常用的 Process 和 非常少用到的 process 出現，會使 `LRU` 的 EAT 降低很多(因為常用的會一直保留在TLB 中)
2. Page Replacement Policy：(**影響Page Fault Rate**)

> 兩者造成的結果都是將不常使用到的踢出 Disk

- 在這個測資中，兩者相差不大
- 增加兩者的差異性

  如果有一些很常被使用到的Page，在 Clock 被踢出的可能性較低，因此 page fault 會降低。

3. Frame Allocation Policy：(**影響Page Fault Rate**)

- 在這個測資中，兩者相差不大
  - 增加兩者的差異性
    - 當某一個 process 一開始就占用到比較多的 frame，那後進來的 process 在 Local 的情況會增加Rate
    - Global 會讓 process 互相影響，local 便不會 (由sim1,sim2看出，每個process 獨立處理自己的memory效率較高)
    - Local 有可能在多個process下，找不到 Frame 可以 allocate