姓名：郭翔恩

學號：M113040033

高等作業系統 作業1報告

## Outline

1. 四種演算法的設計比較：(FIFO, OPT, ESC, MyAlgo)
2. Three test reference strings method：  
    (random pick, locality pick, my pick)
3. 實驗結果  
    Random data(四種演算法的Page fault、Interrupt、Disk write)  
    Locality data(四種演算法的Page fault、Interrupt、Disk write)   
    My pick data(四種演算法的Page fault、Interrupt、Disk write)
4. 結論

#### **一、四種演算法的設計比較：**

先介紹一下四種 page replacement algorithm運作原理，以及我的實作方式

1. First in first out algorithm(FIFO)：  
   先進先出，演算法中最容易實作且現行OS大多都採用的方式，最先放入的會是frame裡最早被替換的。  
   實作是以queue來設計，因為這個資料結構本身就有先進先出的特性。  
   其他輔助的資料結構有dirty bit陣列和是否在記憶體(isInMemory)的陣列(所有演算法都有)。
2. Optimal algorithm(OPT)：  
   未來長期不會參考到的page，作為victim page。Page fault ratio一定是所有演算法中最低的，不會遭遇Belady's Anomaly。但理論上是無法被實作的，因為要看的是未來，本次作業只是先有了資料能看未來才實作的出來。資料結構跟FIFO差不多，queue變成使用vector以及其他輔助資訊的陣列，若未來的值都沒出現，則替換第一個frame。
3. Enhance-Second Chance algorithm(ESC)：  
   以<Reference Bit, Modification Bit>配對值作為挑選victim page的依據，值最小的page視為victim page，若全部page具相同值，則替換第一個index。

<0, 0>：最近沒被行程使用，也沒被行程修改  
<0, 1>：表示最近沒有被行程使用，但有被修改過，須先寫回磁碟後，才可進行替換  
<1, 0>：表示最近曾被行程使用，但是沒被修改過，由於可能再次被使用  
<1, 1>：表示最近曾被行程使用，也被修改過，所以要寫回磁碟中  
實作中使用的資料結構跟前面差不多，只是多了reference bit陣列，邏輯是只要全部reference bit都變1，ESC\_checkReferenceBits這個函數就會把全部reference bit都變0，中斷額外加1。所以實際上我的判斷只有在比<0,0>和<0,1>而已

1. My algorithm(MyAlgo)：  
   我所設計的page replacement演算法是用區段統計的方式，統計1~600這些reference string的出現次數，將來victim page會替換掉出現次數最少的，如果全部的出現次數都一樣，就像OPT一樣，替換掉第一個frame的page。而這些統計的出現次數每1000次會歸0重新開始，會這樣做是因為要給其他人一些機會，因為有的ref str累積太高，但他後面根本不常出現，卻不會被替換掉，這時候常出現的資料反而會一直被替換掉，就會造成部分區段的不公平，資料結構跟OPT差不多，但多了統計1~600出現次數的陣列。

#### **二、Three test reference strings method**

1. Random pick：  
   reference string的選取方式是18萬次每次都random一個值(範圍1~600)。
2. Locality pick：  
   先定好要貢獻的資料長度，我是random 500~1000，一直貢獻直到18萬筆資料為止，接著在15~20個左右數字集合的資料間(作業規定)random反覆挑選，在600個數字中random一個head，範圍: 1到580，我取580的原因是因為萬一取到20，580+20也不會超過600。
3. My pick：  
   我的作法是先用Random pick的方法，18萬筆資料產生之後，隨機1~5步個frame，塞我製造的50個特殊數字Find50SpecificNumber這個函數，也就是這50個數字會穿插在我的隨機數字中，而這50個數字都不一樣，會照順序穿插，50個不一樣的數用完就在輪一次，直到18萬筆資料都塞得差不多。所以對於我的演算法非常有利，因為我用統計的方式可以讓這幾個常出現的數字免於被替換掉的風險。

#### **三、執行結果的探討**

* Random data

圖一為在random pick下的執行結果，可以很明確的觀察出，Optimal algorithm在page fault次數上確實遠遠的小於其他三者，而其他三者在表現上相差不大，不管是FIFO、ESC還是My Algo，圖上可以看到三條幾乎重疊在一起。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 176945 | 174055 | 170961 | 168018 | 164926 | 161918 | 158909 | 155895 | 152934 | 149946 |
| OPT | 153893 | 140512 | 130588 | 122496 | 115508 | 109362 | 103805 | 98737 | 94039 | 89667 |
| ESC | 177005 | 173983 | 170970 | 167966 | 165005 | 161946 | 158941 | 155902 | 152934 | 149943 |
| MyAlgo | 177038 | 174114 | 171100 | 168137 | 165044 | 162028 | 158914 | 155847 | 152929 | 149927 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 221109 | 217471 | 213751 | 209923 | 205952 | 202504 | 198700 | 194854 | 190850 | 187466 |
| OPT | 192095 | 175236 | 163122 | 152941 | 144426 | 136759 | 129820 | 123657 | 117416 | 111969 |
| ESC | 238613 | 226243 | 219629 | 214414 | 209750 | 205330 | 201270 | 197310 | 192884 | 189138 |
| MyAlgo | 221544 | 217770 | 213687 | 210270 | 206185 | 202479 | 198814 | 194735 | 191351 | 187629 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 44164 | 43416 | 42790 | 41905 | 41026 | 40586 | 39791 | 38959 | 37916 | 37520 |
| OPT | 38202 | 34724 | 32534 | 30445 | 28918 | 27397 | 26015 | 24920 | 23377 | 22302 |
| ESC | 43744 | 43406 | 42807 | 42098 | 41296 | 40536 | 39910 | 39310 | 38103 | 37550 |
| MyAlgo | 44506 | 43656 | 42587 | 42133 | 41141 | 40451 | 39900 | 38888 | 38422 | 37702 |

* Locality data

這部分由於我區間範圍設置的比較大(500~1000次可能差不多資料)，所以從10個frame以後pagefault會明顯降低，因為是15~20個數字集合內的數字重複，20個frame很可能包含全部，跳到下個區間才有可能才開始有pagefault，而我的演算法(統計)frame太少很難展現統計的優勢，邏輯上也是一直替換第一個(很有可能大家total一樣，常出現的因為在第一個一直被替換掉)，所以比不過其他演算法，而ESC稍微優於FIFO，可能是因為有reference bit的關係，常出現的就不容易被替換掉。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 82101 | 5504 | 4332 | 4277 | 4211 | 4114 | 4008 | 3862 | 3791 | 3737 |
| OPT | 40053 | 4591 | 3802 | 3449 | 3193 | 2981 | 2816 | 2676 | 2537 | 2419 |
| ESC | 82173 | 5783 | 4524 | 4235 | 4182 | 4048 | 3951 | 3856 | 3742 | 3645 |
| MyAlgo | 110958 | 55339 | 26161 | 7553 | 4926 | 4040 | 3973 | 3903 | 3836 | 3752 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 102588 | 6892 | 5421 | 5347 | 5252 | 5144 | 4968 | 4778 | 4698 | 4621 |
| OPT | 50013 | 5743 | 4695 | 4304 | 4041 | 3719 | 3445 | 3332 | 3146 | 2980 |
| ESC | 115874 | 8316 | 5845 | 5420 | 5338 | 5158 | 4980 | 4868 | 4695 | 4568 |
| MyAlgo | 138732 | 69147 | 32539 | 9463 | 6140 | 5022 | 4923 | 4870 | 4761 | 4662 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 20487 | 1388 | 1089 | 1070 | 1041 | 1030 | 960 | 916 | 907 | 884 |
| OPT | 9960 | 1152 | 893 | 855 | 848 | 738 | 629 | 656 | 609 | 561 |
| ESC | 20745 | 1472 | 1089 | 1053 | 1043 | 1031 | 959 | 955 | 901 | 879 |
| MyAlgo | 27774 | 13808 | 6378 | 1910 | 1214 | 982 | 950 | 967 | 925 | 910 |

* My pick data

這是一個random的情境但又一直有重複出現的數字穿插在字串，所以我的演算法表現得特別好(專門處理這種類型的reference string)，而ESC到後面也稍微優於FIFO，畢竟ESC的概念也類似最近常出現的盡量不要換掉，而FIFO一視同仁，所以就沒太卓越的表現。OPT則依然是天花板。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 177305 | 174603 | 171895 | 169220 | 166518 | 163772 | 161477 | 160096 | 158895 | 157732 |
| OPT | 154950 | 140513 | 127808 | 115646 | 103951 | 93199 | 84889 | 78582 | 73438 | 69036 |
| ESC | 177303 | 174611 | 171946 | 169238 | 166640 | 163708 | 160740 | 157066 | 151247 | 145096 |
| MyAlgo | 173006 | 164578 | 154877 | 142364 | 127737 | 120300 | 115910 | 112434 | 110014 | 107110 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 221402 | 218058 | 214955 | 211486 | 208089 | 205041 | 201874 | 199973 | 198595 | 197193 |
| OPT | 193461 | 175774 | 159863 | 144378 | 129824 | 116090 | 106172 | 98050 | 91971 | 86512 |
| ESC | 239454 | 226982 | 220734 | 215874 | 211998 | 207429 | 203352 | 198245 | 191014 | 182886 |
| MyAlgo | 216514 | 205563 | 193806 | 178354 | 159529 | 150435 | 144797 | 140185 | 137707 | 133843 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algo\frame | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| FIFO | 44097 | 43455 | 43060 | 42266 | 41571 | 41269 | 40397 | 39877 | 39700 | 39461 |
| OPT | 38511 | 35261 | 32055 | 28732 | 25873 | 22891 | 21283 | 19468 | 18533 | 17476 |
| ESC | 44278 | 43504 | 42921 | 42268 | 41893 | 40855 | 40176 | 39064 | 37903 | 36128 |
| MyAlgo | 43508 | 40985 | 38929 | 35990 | 31792 | 30135 | 28887 | 27751 | 27693 | 26733 |

#### **四、結論**

OPT依舊是所有資料類型的天花板，即便是對我演算法有利的字串，依然不敵OPT。而FIFO跟ESC以及自己的演算法在一般隨機的情況下其實差不多，只是在常常出現重複資料的case下，FIFO就相對沒什麼優勢，但FIFO在實作比較簡單還是一個潛在優勢。