Advanced Operating Systems - Homework 1

學號:M093040003

姓名:趙云珮

2020年10月26日

1 Three test reference strings

1.1 Random

從1～800中，隨機取一個數字當作起始數字,再從[1，25]連續數字中隨機取一個數字當作區間，將起始數字加上區間數字，此範圍內數字作爲reference string，重複此動作直到總共取到200000筆數字。

1.2 Locality

從1～800中，隨機取一個數字當作起始數字,再從[1/25，1/15]範圍內隨機取一個數乘以800作爲範圍，再平分這個區間，並對起始數字做正負，reference string即爲起始數字的正負平分的範圍，重複此動作直到總共取到200000筆數字。

1.3 My own reference string

從1～800中，隨機取一個數字當作起始數字,再從[1/25，1/15]範圍內隨機取一個數乘以800作爲區間，將起始數字加上區間數字，此範圍內數字作爲reference string，再將此範圍內數字取亂數值到達到共8000筆數字，再重複此動作直到總共取到200000筆數字。我的reference string改良上述Locality作法，重複選取先前已取到的數字，用以達到更高的Locality 性質。

2 Page Replacement Algorithms

2.1 FIFO algorithm

FIFO的作法是先進先出，在這裡我的實作是使用queue。page fault的計算是當queue內爲空或是未滿或需進行替換page時，分別需計算一次。而interrupt部分則是當查詢或存取queue或修改，就計算一次。至於disk I/O部分，是當page fault 產生或dirty bit有修改過，就計算一次。

2.2 ARB algorithm (8-bit information)

ARB的作法是利用reference bits紀錄每個page，定期將reference bits轉換爲高階位，即將其他位右移，降低最低位。在這裡我的實作是使用二進位的reference bits轉換成十進位來進行原ARB shift的方式。page fault的計算是當page table未滿或需進行替換page時，需計算一次。而interrupt部分則是當查詢或存取page table或修改，就計算一次。至於disk I/O部分，是當page fault產生或dirty bit有修改過，就計算一次。

2.3 Enhanced second-chance algorithm

ESC的作法是使用兩個bits來決定victim page的替換順序，排序爲:(0,0)，(0,1)，(1,0)，(1,1)，第一個bit 爲reference bits，第二個則爲dirty bit，在計算interrupt時，只要有修改過這兩個bit，便計算一次，在page fault及disk I/O 則如同ARB計算方式。

2.4 My own algorithm

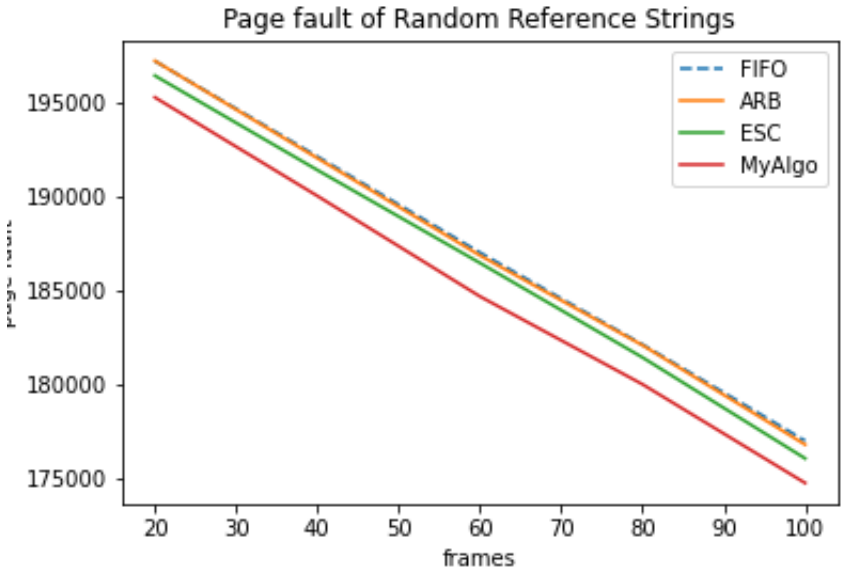
我的作法是使用hash，利用Knuth's multiplicative 的integer hash 公式，該公式爲:

$$\text{hash(i)} = i * 2654435761 mod 2^{32}$$

而我將原公式的2的32次改爲符合我的page table 的大小，經畫圖驗證，page fault 是有小於FIFO，而interrupt及disk I/O 則同ARB計算方式。

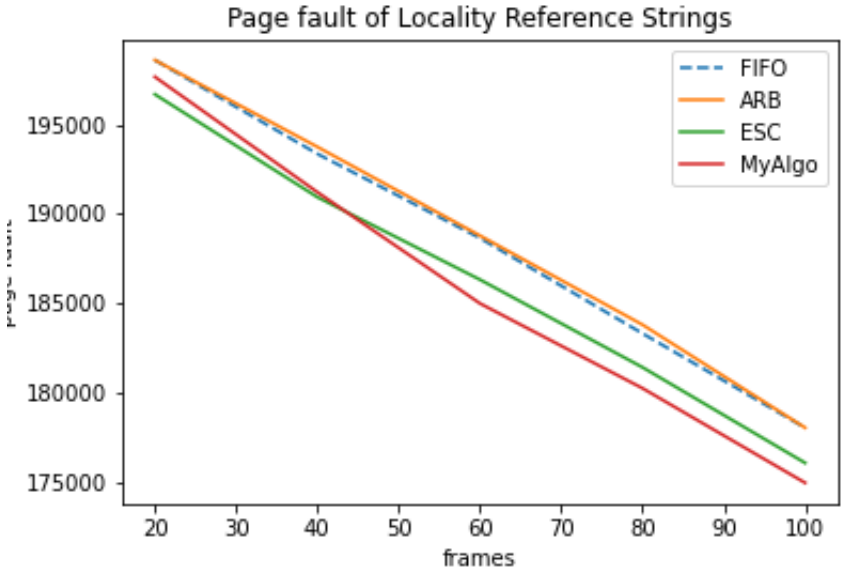
3 The relationship

3.1 Page faults and the number of frames



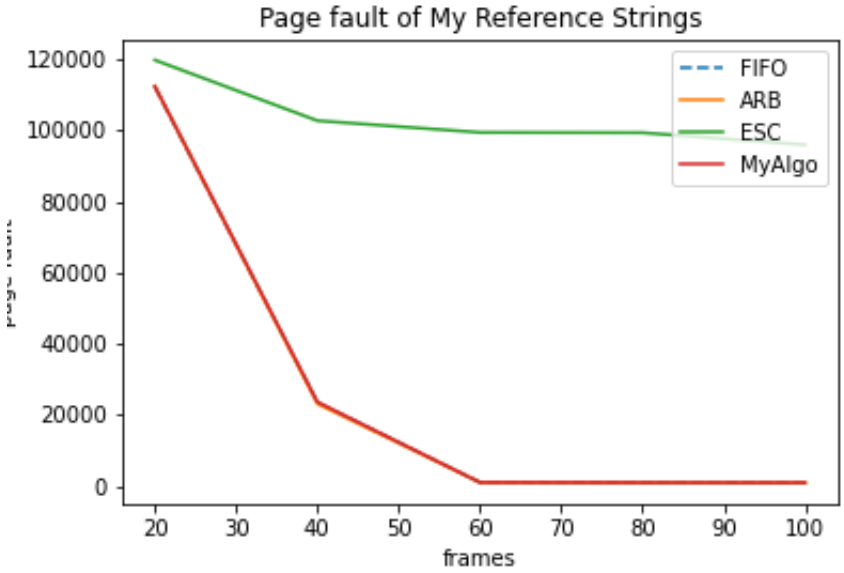
Reference string在Random的情況下，所有的演算法皆隨著frame的數量增大，page fault會下降，且無出現Belady's Anomaly。

比較這四種演算法，我的演算法勝過其他三種，其他的演算法效果差不多。



Reference string在Locality的情況下，所有的演算法皆隨著frame的數量增大，page fault會下降，且無出現Belady's Anomaly。

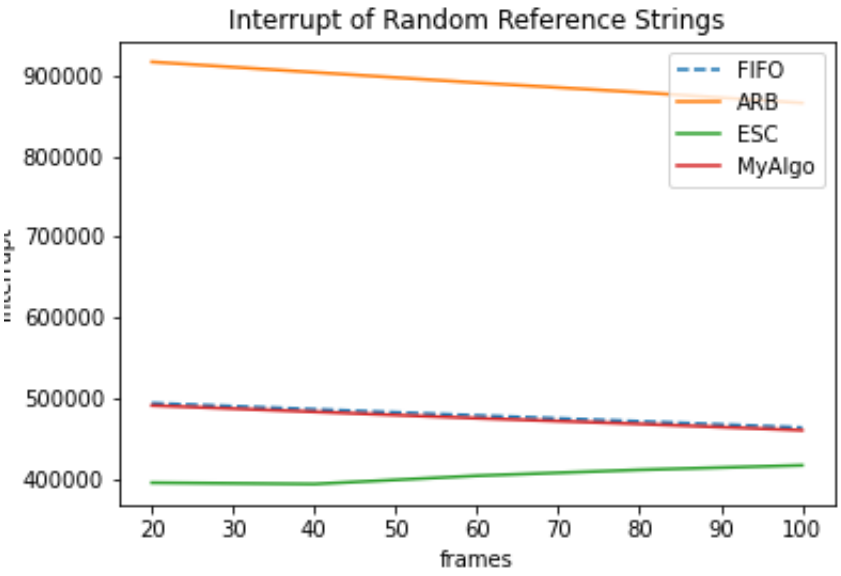
與Random不同之處，比較這四種演算法，在剛開始我的演算法略輸於ESC演算法，但在大約40個frames的時候，開始勝過ESC，而其他兩種演算法效果差不多。



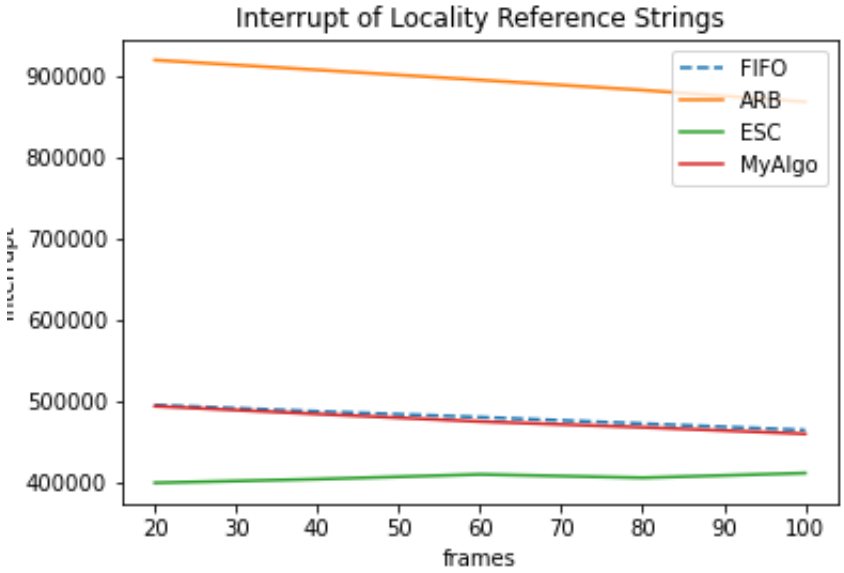
Reference string在我自己設計的情況下，所有的演算法皆隨著frame的數量增大，page fault會下降，且無出現Belady's Anomaly。

我的演算法相對其他的演算法，page fault數量急遽下降，當大約60個frames的時候，page fault數量便無明顯變化。

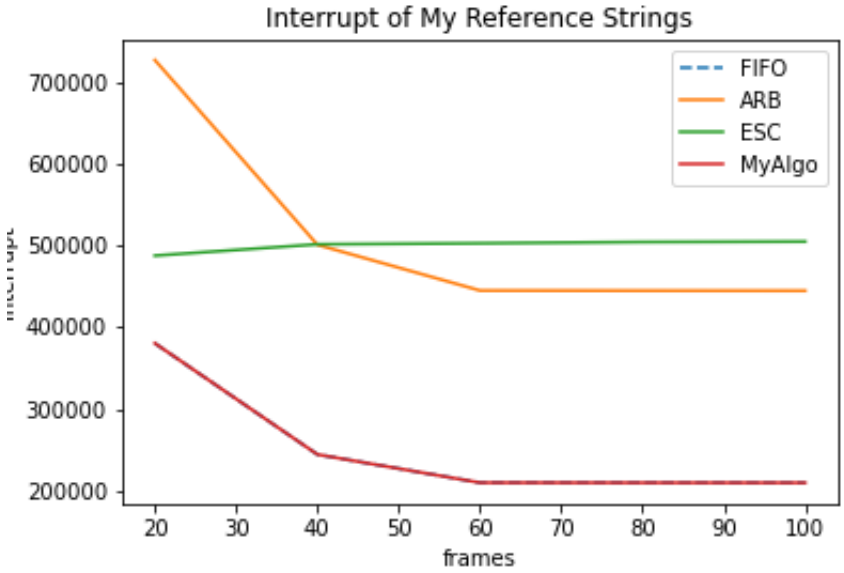
3.2 The number of interrupts and the number of frames



Reference string在Random的情況下，演算法隨著frame的數量增大，interrupt會下降，而ARB演算法的interrupt數量一直遠高於其他三種是因爲此演算法使用到reference bit，週期部分我以8 bit爲一次，因此進行相當多的interrupt。我的演算法與FIFO演算法的效果差不多。

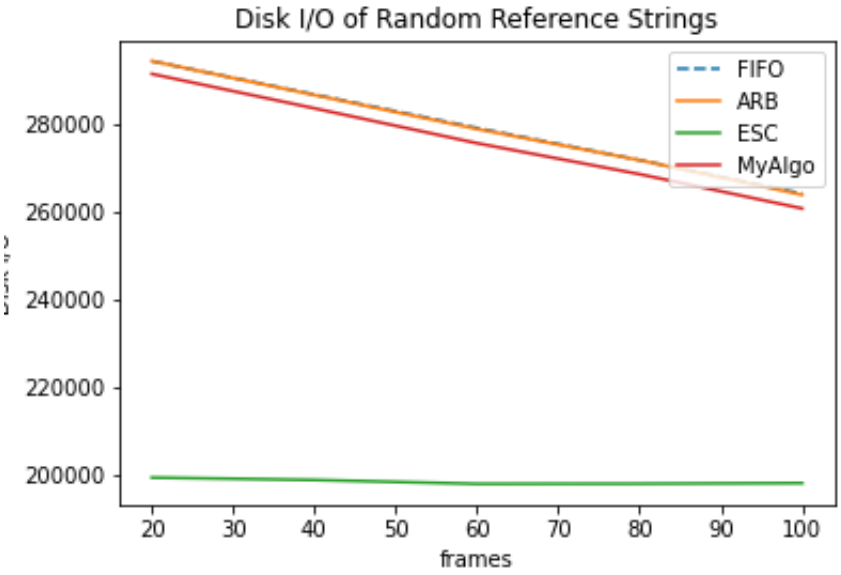


Reference string在Locality的情況下，演算法隨著frame的數量增大，interrupt會下降，與Random不同的是，ESC在60個frames的時候，interrupt不會逐漸上升，是因爲Locality 在一定範圍內出現的reference string 較接近，所以需要更動的 page 較少。

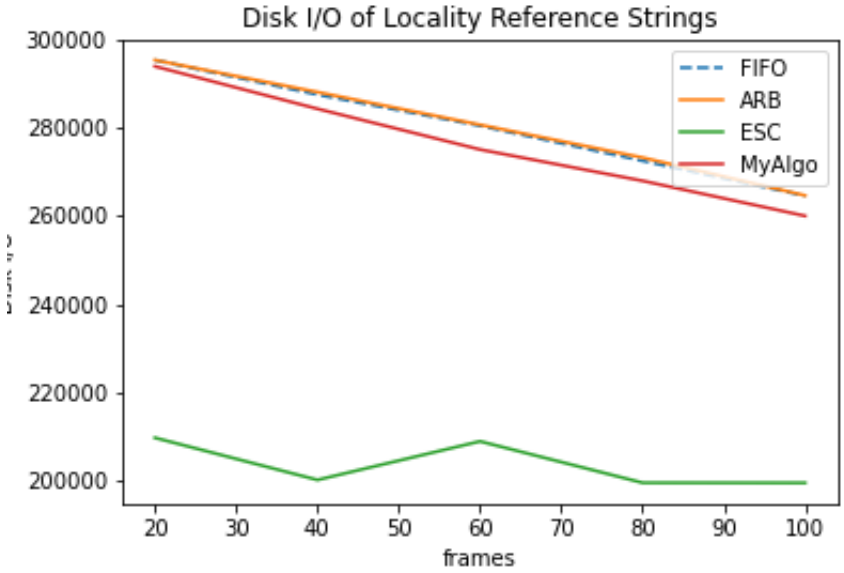


Reference string在我自己設計的情況下，演算法隨著frame的數量增大，interrupt會下降，ARB演算法的interrupt仍然是偏高的。而我的演算法的interrupt近似於FIFO演算法。

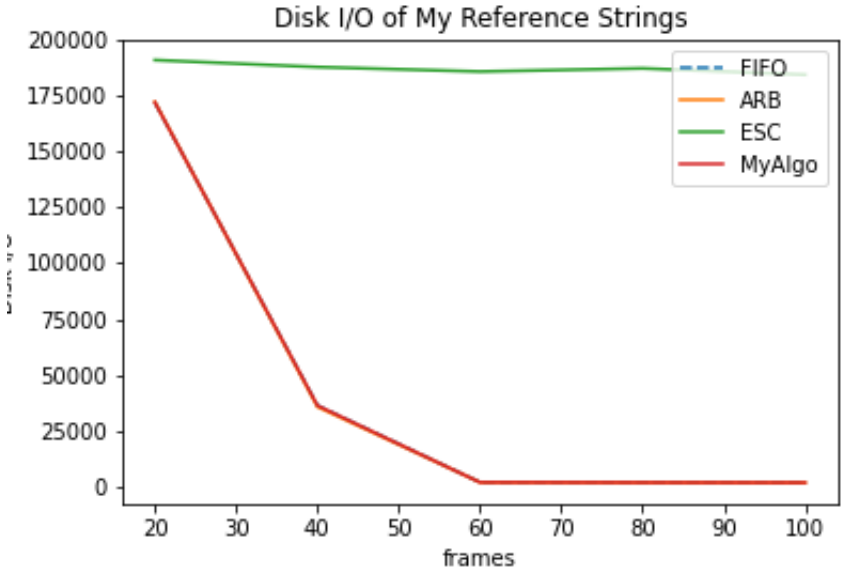
3.3 The number of disk writes (in pages) and the number of frames



Reference string在Random的情況下，演算法隨著frame的數量增大，Disk I/O會下降，計算次數與page fault相同，但是當dirty bit要修改的時候會再額外計算一次。FIFO演算法的Disk I/O效果與ARB演算法差不多。



Reference string在Locality的情況下，演算法隨著frame的數量增大，Disk I/O會下降，與Random不同的是，我的演算法在開始的時候與ARB及FIFO演算法差不多，但之後下降程度比這兩種演算法多。



Reference string在Locality的情況下，演算法隨著frame的數量增大，Disk I/O會下降，而ESC演算法由於會因爲dirty bit的關係，進行很多的Disk I/O，因此數量遠多於其他演算法。