OS Project 說明文件

資訊三甲 11020136 鄭絜馨

1. 開發環境

   Visual Studio Code(python)

2. 實作方法和流程

- 資料結構

  將讀入的 Page Reference 的次序存在一個 list，每單位時間內存的 page frame 內容以 class 型別儲存，記錄當前的 reference 的值、page frame 值、是否有 page fault。

- FIFO

  用 for 迴圈執行每個要放入 page frame 的值，若此資料不在 page frame 內且當前 page frame 使用的空間小於 page frame 的個數，直接將此資料放入 page frame，若 page frame 使用的空間等於 page frame 的個數將最先進到 page frame 內的值 pop 掉，再將新的資料放入，以上兩種情況 page fault 會設為 F，若此資料已經在 page frame 內則不做更動，判斷三種情況後會將最後 page frame 的狀態以 class 型別存到 list 完成一次的迴圈。

- LRU

  結構與 FIFO 類似，用 for 迴圈執行每個要放入 page frame 的值，若此資料不在 page frame 內且當前 page frame 使用的空間小於 page frame 的個數，直接將此資料放入 page frame，若 page frame 使用的空間等於 page frame 的個數將最先進到 page frame 內的值 pop 掉，再將新的資料放入，以上兩種情況 page fault 會設為 F，若要加入的資料存在於 page frame 要將他換到 list 最後，代表最近有使用到，才不會在 page frame 滿的時候優先被替換掉。

- LFU+FIFO

  結構也與 FIFO 類似，用 for 迴圈執行每個要放入 page frame 的值，若此資料不在 page frame 內且當前 page frame 使用的空間小於 page frame 的個數，直接將此資料放入 page frame，但是當 page frame 滿的時候會優先替換掉出現頻率最小的。以 dictionary 記錄每筆資料出現的頻率，若該資料被 pop 掉則頻率設為 0，若該資料已存在在 page frame 則該資料頻率會加 1，否則在加入 page frame 時將該資料頻率設為 1，當 page frame 滿了會選擇在 page frame 裡且頻率最小的 pop 出

去，以 function (find_smallest_counter)找出當前 page frame 內頻率最小的。

- MFU+FIFO

  大致上與 LFU+FIFO 一樣，用 for 迴圈執行每個要放入 page frame 的值，若此資料不在 page frame 內且當前 page frame 使用的空間小於 page frame 的個數，直接將此資料放入 page frame，當 page frame 滿的時候會優先替換掉出現頻率最大的。以 dictionary 記錄每筆資料出現的頻率，若該資料被 pop 掉則頻率設為 0，若該資料已存在在 page frame 則該資料頻率會加 1，否則在加入 page frame 時將該資料頻率設為 1，當 page frame 滿了會選擇在 page frame 裡且頻率最大的 pop 出去，以 function(find_largest_counter)找出當前 page frame 內頻率最大的。

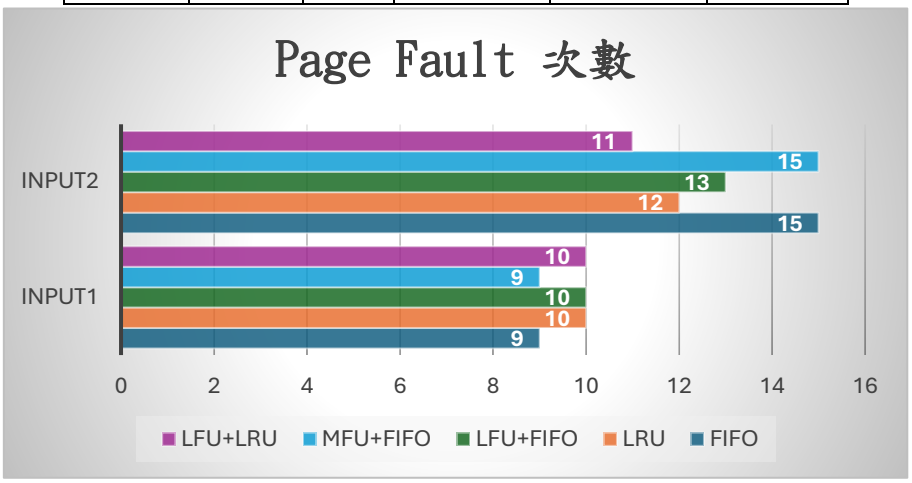
- LFU+LRU

  結合 LFU&LRU，主體主要是用 LFU，用 for 迴圈執行每個要放入 page frame 的值，若此資料不在 page frame 內且當前 page frame 使用的空間小於 page frame 的個數，直接將此資料放入 page frame，但是當 page frame 滿的時候會優先替換掉出現頻率最小的。以 dictionary 記錄每筆資料出現的頻率，若該資料被 pop 掉則頻率設為 0，若該資料已存在在 page frame 則該資料頻率會加 1，且將他的位置移動到 list 最後面，讓下次要被替換的是最久沒被使用到的頁，否則在加入 page frame 時將該資料頻率設為 1，當 page frame 滿了會選擇在 page frame 裡且頻率最小的 pop 出去，以 function (find_smallest_counter)找出當前 page frame 內頻率最高的。

3. 不同方法比較

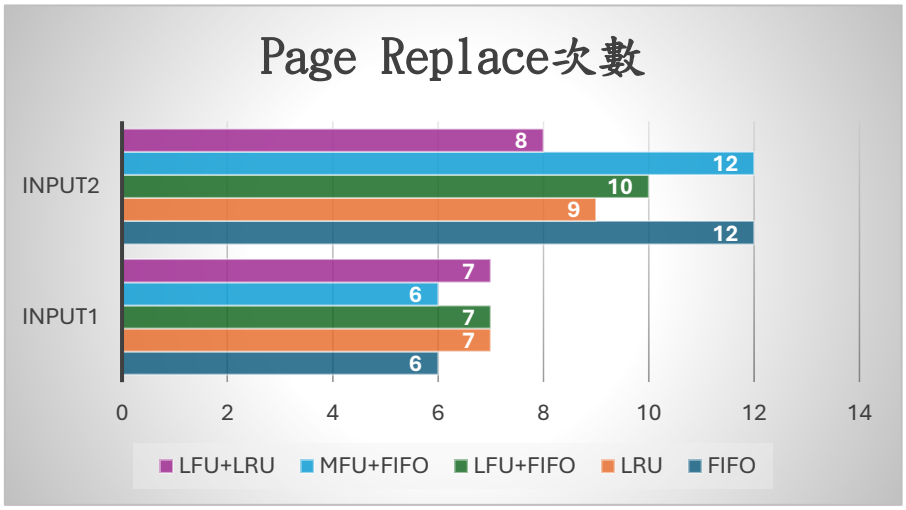
- Page Fault 次數

	FIFO	LRU	LFU+FIFO	MFU+FIFO	LFU+LRU
input1	9	10	10	9	10
input2	15	12	13	15	11



- Page Replace 次數

	FIFO	LRU	LFU+FIFO	MFU+FIFO	LFU+LRU
input1	6	7	7	6	7
input2	12	9	10	12	8



由上表可知 page fault 發生次數必定大於 page replace 次數，因為發生 page fault 不一定會發生 page replace，若 page frame 還沒塞滿就不會發生 page replace。

使用 FIFO 發生 page fault 的次數大於使用 LRU，因為 LRU 會根據最近有無被參考來排序，而 FIFO 僅僅根據頁面進入的時間順序進行替換，不考慮頁面的實際使用情況，這樣可能會替換掉實際上仍頻繁被訪問的頁面。

4. 結果與討論

- 實驗數據與發現

A. FIFO 與 LRU 的 page fault 次數在不同輸入下表現不一，而 LRU 的表現通常更好，這意味著 LRU 更能夠適應真實的訪問模式。

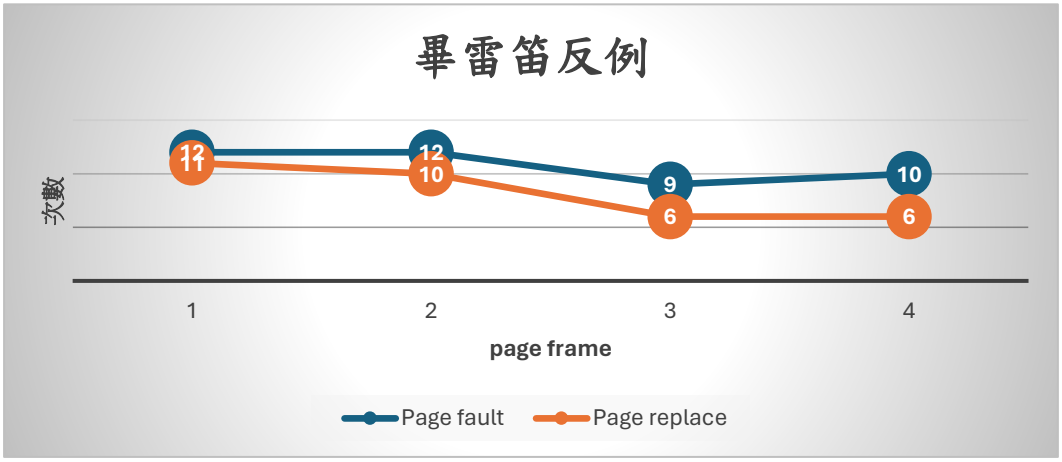
B. LFU 和 MFU 的策略在某些情況下有效，但可能會在其他情況下導致更多的 page fault。

C. LFU+LRU 在大多數情況下表現最好，因為它結合了頻率和最近使用的優點。

- 畢雷笛反例

以 FIFO 為例，Reference string: 123412512345

Page frame	1	2	3	4
Page fault	12	12	9	10
Page replace	11	10	6	6



由此可知增加頁框數不一定會減少 page fault 發生，但這是少數特例，大部分情況還是會在 frame 數增加時減少 page fault 發生的次數。