資訊三甲 10827145 黃霈昕 作業系統 HW3 document

1. 開發環境

   CPU: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
   HDD:931GB
   RAM:8GB
   OS: Windows 10 - 21H1
   系統類型:64 位元作業系統，x64 型處理器
   程式編輯器: Visual Studio Code
   程式語言:python(3.9)

2. 實作方法和流程

   *記錄 Page frame 的 List 中 Page 由新到舊排序

   FIFO: 讀入指定檔並將所有 Page Reference String 存成一個 String，然後對每個 Page 進行 Page in、Page out。若要載入的 Page 已存在於 Page frame 當中，就直接將 Page number 及 Page frame 輸出到 output 檔。若要載入的 Page 不存在於 Page frame 當中，即為 Page Fault，接著就要判斷是否需要 Page Replace，若 Page frame 滿了就要 Page Replace，將最早被載入 Page frame 的 Page 做 Page out(也就是從記錄 Page frame 的 List 移除)然後將該 Page 做 Page in(也就是插入到記錄 Page frame 的 List 的最前面)；若 Page frame 沒滿就直接將該 Page 做 Page in；然後，將 Page number、Page frame 及表示 Page Fault 的 F 輸出到 output 檔。最後印出 Page Fault、Page Replaces、Page Frames 到 output 檔。

   LRU: 讀入指定檔並將所有 Page Reference String 存成一個 String，然後對每個 Page 進行 Page in、Page out。若要載入的 Page 已存在於 Page frame 當中，就將該 Page 移到記錄 Page frame 的 List 的最前面，然後將 Page number 及 Page frame 輸出到 output 檔。若要載入的 Page 不存在於 Page frame 當中，即為 Page Fault，接著就要判斷是否需要 Page Replace，若 Page frame 滿了就要 Page Replace，將最近沒被用到的 Page 做 Page out(也就是從記錄 Page frame 的 List 移除)然後將該 Page 做 Page in(也就是插入到記錄 Page frame 的 List 的最前面)；若 Page frame 沒滿就直接將該 Page 做 Page in；然後，將 Page number、Page frame 及表示 Page Fault 的 F 輸出到 output 檔。最後印出 Page Fault、Page Replaces、Page Frames 到 output 檔。

   LFU+ LRU: 讀入指定檔並將所有 Page Reference String 存成一個 String，然後對每個 Page 進行 Page in、Page out。若要載入的 Page 已存在於 Page frame 當中，就將該 Page 移到記錄 Page frame 的 List 的最前面，並將記錄該 Page 的

counter 加一，然後將 Page number 及 Page frame 輸出到 output 檔。若要載入的 Page 不存在於 Page frame 當中，即為 Page Fault，接著就要判斷是否需要 Page Replace，若 Page frame 滿了就要 Page Replace，將 counter 值最小且最近沒被用到的 Page 做 Page out(也就是從記錄 Page frame 的 List 移除)然後將該 Page 做 Page in(也就是插入到記錄 Page frame 的 List 的最前面，counter 從 0 開始)，並將記錄該 Page 的 counter 加一；若 Page frame 沒滿就直接將該 Page 做 Page in(counter 從 0 開始)，並將記錄該 Page 的 counter 加一；然後，將 Page number、Page frame 及表示 Page Fault 的 F 輸出到 output 檔。最後印出 Page Fault、Page Replaces、Page Frames 到 output 檔。

MFU+FIFO: 讀入指定檔並將所有 Page Reference String 存成一個 String，然後對每個 Page 進行 Page in、Page out。若要載入的 Page 已存在於 Page frame 當中，就把該 Page 的 counter 加一然後直接將 Page number 及 Page frame 輸出到 output 檔。若要載入的 Page 不存在於 Page frame 當中，即為 Page Fault，接著就要判斷是否需要 Page Replace，若 Page frame 滿了就要 Page Replace，將 counter 值最大且最早被載入 Page frame 的 Page 做 Page out(也就是從記錄 Page frame 的 List 移除)然後將該 Page 做 Page in(也就是插入到記錄 Page frame 的 List 的最前面，counter 從 0 開始) ，並將記錄該 Page 的 counter 加一；若 Page frame 沒滿就直接將該 Page 做 Page in(counter 從 0 開始)，並將記錄該 Page 的 counter 加一；然後，將 Page number、Page frame 及表示 Page Fault 的 F 輸出到 output 檔。最後印出 Page Fault、Page Replaces、Page Frames 到 output 檔。

MFU+LRU: 讀入指定檔並將所有 Page Reference String 存成一個 String，然後對每個 Page 進行 Page in、Page out。若要載入的 Page 已存在於 Page frame 當中，就將該 Page 移到記錄 Page frame 的 List 的最前面，並將記錄該 Page 的 counter 加一，然後將 Page number 及 Page frame 輸出到 output 檔。若要載入的 Page 不存在於 Page frame 當中，即為 Page Fault，接著就要判斷是否需要 Page Replace，若 Page frame 滿了就要 Page Replace，將 counter 值最大且最近沒被用到的 Page 做 Page out(也就是從記錄 Page frame 的 List 移除)然後將該 Page 做 Page in(也就是插入到記錄 Page frame 的 List 的最前面，counter 從 0 開始)，並將記錄該 Page 的 counter 加一；若 Page frame 沒滿就直接將該 Page 做 Page in(counter 從 0 開始)，並將記錄該 Page 的 counter 加一；然後，將 Page number、Page frame 及表示 Page Fault 的 F 輸出到 output 檔。最後印出 Page Fault、Page Replaces、Page Frames 到 output 檔。

3. 不同方法的比較

使用 input2

Page Fault

	Page Frames = 3	Page Frames = 4	Page Frames = 5
FIFO	15	10	9
LRU	12	8	7
LFU+ LRU	11	9	7
MFU+FIFO	15	12	8
MFU+LRU	12	9	8

Page Replaces

	Page Frames = 3	Page Frames = 4	Page Frames = 5
FIFO	12	6	4
LRU	9	4	2
LFU+ LRU	8	5	2
MFU+FIFO	12	8	3
MFU+LRU	9	5	3

FIFO: FIFO 沒有考慮 Page 被 reference 到的最新時間，而是直接以載入 Page Frame 的時間點將最舊的 Page 與要載入的 Page 做置換，可能把會用到的 Page 置換掉，因此在 input2 中 Page Fault 和 Page Replace 次數都比 LRU 高，效能較差。

LRU: LRU 會依被 reference 到的最新時間進行排序的動作，將最久沒用的 Page 與要載入的 Page 做置換，因此在 input2 中 Page Fault 和 Page Replace 次數都比 FIFO 少，效能較佳。

LFU+ LRU: LFU+ LRU 不僅考慮到了 Page 被 reference 到的最新時間也考慮了近期內 Page 被 reference 的頻率，經常使用相同 Page 的情況下會有較高的效率，因此在 input2 中 Page Fault 和 Page Replace 次數都較少，有較好的表現。

MFU+FIFO: MFU+FIFO 沒有考慮 Page 被 reference 到的最新時間且將近期被 reference 頻率高的 Page 與要載入的 Page 做置換，雖然常用不代表正在用，但在經常使用相同 Page 的情況下會導致效能降低，因此在 input2 中 Page Fault 和 Page Replace 次數都較多，表現較差。

MFU+LRU: MFU+LRU 會依被 reference 到的最新時間進行排序的動作，將近期內 Page 被 reference 的頻率較高且最久沒用的 Page 與要載入的 Page 做置換，雖然考慮到了 Page 被 reference 到的最新時間，但在經常使用相同 Page 的情況下會導致效能降低，因此在 input2 中 Page Fault 和 Page Replace 次數都較 MFU+FIFO 少，但也較其他方法多，表現比 MFU+FIFO，但還是不佳。

在 input2 中 FIFO 沒有考慮 Page 被 reference 到的最新時間，導致表現不佳，效能較 LRU 差，但 LRU 需一直做排序，在實際使用上可能會較複雜，兩者各有利弊。
雖然在 input2 中 LFU 比 MFU 有效率，但有時候常用不代表正在用，有可能在某些時刻集中使用，但多數時間沒有使用，因此這種時候使用 MFU 反而會比 LFU 有效率，LFU 和 MFU 皆無法保證表現一定較佳。

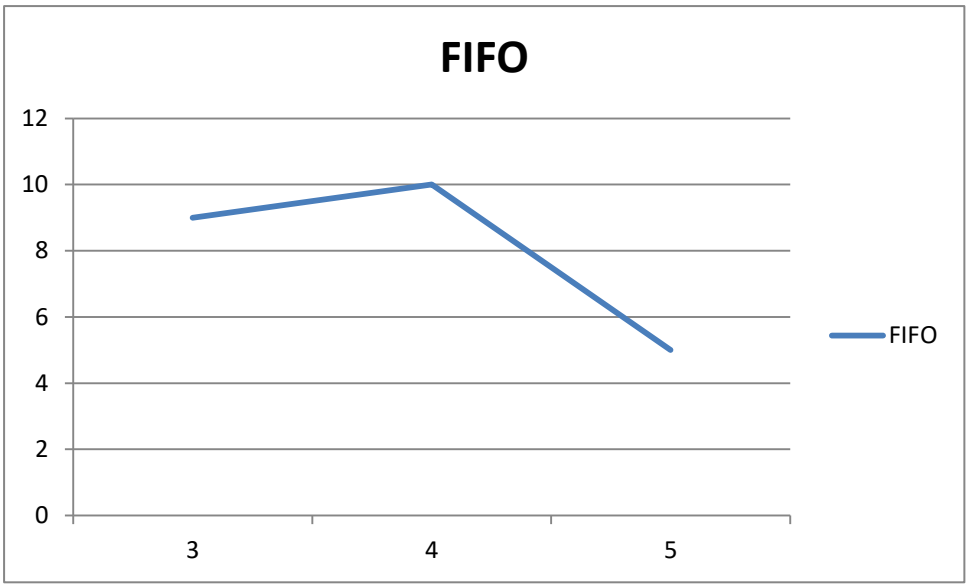
4. 結果與討論
在 input2 中，所有方法都是 Page Frame 越多 Page Fault 越少，但是在有些情況下 Page Frame 增加並不一定會使 Page Fault 減少，出現少數的特例--畢雷笛反例。

使用 input1
Page Fault

	Page Frames = 3	Page Frames = 4	Page Frames = 5
FIFO	9	10	5
LRU	10	8	5
LFU+ LRU	10	8	5
MFU+FIFO	9	10	5
MFU+LRU	9	10	5

Page Replaces

	Page Frames = 3	Page Frames = 4	Page Frames = 5
FIFO	6	6	0
LRU	7	4	0
LFU+ LRU	7	4	0
MFU+FIFO	6	6	0
MFU+LRU	6	6	0



在 input1 中出現了畢雷笛反例，使用 FIFO 時 Page Frame 增加了但 Page Fault 反而增加了，因為沒有考慮 Page 被 reference 到的最新時間，而 Page Frame 的數量剛好使在 Page 被置換掉後很快又需要被 reference，導致了畢雷笛反例。
使用 LRU 先被放入的元素一定會先被看到，若增加 Page Frame，也只是在原 Page 的次序後面增加一些 Page 而已，因此 Page Frame 增加了 Page Fault 一定會減少，而不會有畢雷笛反例。
在 input1 中除了 FIFO 外，MFU+FIFO 和 MFU+LRU 也出現 Page Frame 增加 Page Fault 反而增加了的情況。 MFU+LRU 雖然沒有使用 FIFO，但依據近期內 Page 被 reference 的頻率較高置換了一些 Page，導致 LRU 先被放入的元素一定會先被看到的優勢沒有了，因此無法保證若增加 Page Frame，也只是在原 Page 的次序後面增加一些 Page 而已，而有了 Page Frame 增加 Page Fault 反而增加了的情況。