Advanced Operating Systems

Homework 1

Page Replacement Algorithms and Evaluation

學號：M133040007 | 姓名：李昱佑 | 信箱：denny99203@gmail.com

目錄

1. 實驗設置

1.1 Reference String

依照題目要求，本實驗的 reference string 的範圍設定為 1 至 1200，長度為 120,000。而每個 page 的 dirty bit 為隨機決定。

1.1.1 Random

Random reference string 中的每一個 reference 全部皆是隨機挑選。

1.1.2 Locality

Locality reference string 則是會模擬 procedure calls。而每個 procedure calls 的 reference string 長度會佔總長度的$\frac{1}{200}$～$\frac{1}{100}$。並且符合題目要求，每個 procedure calls 的長度是隨機的。而每一個 procedure calls 的 page number 範圍則是隨機挑選一個連續的 page number 範圍。

由於 locality reference string 中是模擬連續的 procedure calls，所以 reference string 是由各個不同長度的 procedure calls 所組成。

1.1.3 Hybrid

這是我所設計的 reference string，這是將 Random 與 Locality 的生成方法結合在一起。Hybrid reference string 由 Random 與 Locality 兩方法輪流生成。其中，每一個方法每一次所佔的長度皆是隨機的。

設計 Hybrid reference string 的原因是為了更真實地模擬實際應用場景，因為在現實中，reference string 通常不會只不斷地呼叫 procedure，也不會完全隨機。因此，透過結合這兩種方法，我認為 Hybrid reference string 能夠更貼近真實的使用情境。

1.2 Page Replacement Algorithm

1.2.1 FIFO algorithm

在發生 page fault 時，FIFO (First-In, First-Out) 演算法會將記憶體中最早進入的 page 作為替換對象。換句話說，它只根據 page 進入記憶體的順序來決定替換對象，而不考慮該 page 最近是否被使用過。因此，最早載入的 page 將會優先被替換，即使該 page 仍可能是頻繁被存取的。

1.2.2 Optimal algorithm

在發生 page fault 時，Optimal 演算法會選擇未來最長時間不會被使用的 page 作為替換對象。這種方法能夠在理論上達到最少的 page fault 數，因為它總是預測最理想的替換選擇。

1.2.3 ESC algorithm

Enhanced Second-Chance 演算法是 Second-Chance 演算法的改良版本。它使用了兩個位元來決定要替換的 page： reference bit 和 dirty bit。當發生 page fault 時，演算法會根據這兩個位元的組合來決定哪一個 page 應該被替換。

1.2.4 LFU-DA algorithm

LFU-DA 全名為 Least Frequently Used with Dirty-Aware Aging，此演算法為我透過優化 LFU 所設計的演算法。在這個演算法中，它使用 counter 來追蹤每個 page 的使用頻率。它與傳統的 LFU 一樣，系統都會透過 timer 定期發出 interrupt，讓每個 page 的 counter 定期衰減，以避免某些已經不再被頻繁使用的 page 長期留在記憶體中。

然而，我的方法與 LFU 的不同之處在於它在降低每個 page 的 counter 時考慮了 dirty bit。具體來說，當 page 的 counter 進行衰減時，如果 page 的 dirty bit 為 1，則將 counter 右移 1 位元(相對於除以 2)；如果 dirty bit 為 0，counter 右移 2 位元(相對於除以 4)。

透過這樣的設計，我希望可以減少不必要的 disk writes。換句話說，我們希望在替換 page 時，優先考慮不需要寫回磁碟的 page，從而提升系統效能。

2. 實驗結果

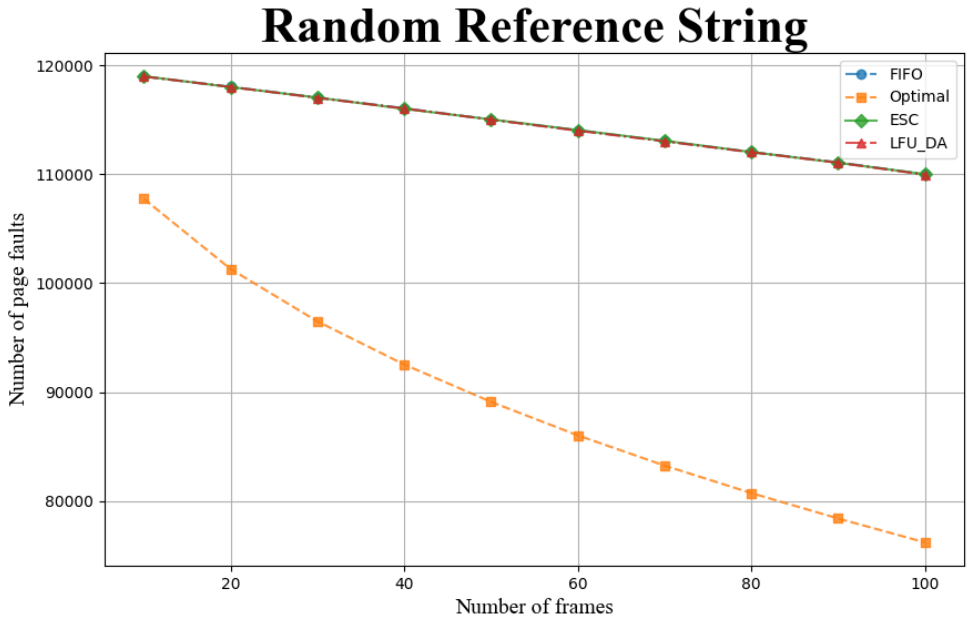
2.1 在 Random reference string 中比較

表一：使用 Random reference string 時各個演算法的表現

Frame Count	FIFO			Optiaml			ESC			LFU-DA		
	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes
10	118991	178422	59431	107766	161610	53844	119013	178341	59328	118989	179617	59428
20	118046	176997	58951	101307	151883	50576	118017	176729	58712	118028	178168	58940
30	117055	175492	58437	96497	144688	48191	117056	175140	58084	117023	176635	58412
40	116074	174025	57951	92535	138750	46215	116015	173445	57430	116051	175167	57916
50	115029	172440	57411	89101	133601	44500	115048	171872	56824	115020	173582	57362
60	114052	170980	56928	86025	128952	42927	114063	170286	56223	114009	172061	56852
70	113054	169497	56443	83244	124754	41510	113082	168657	55575	113030	170600	56370
80	112049	167981	55932	80719	120986	40267	112050	166969	54919	112030	169105	55875
90	111103	166565	55462	78383	117458	39075	111063	165389	54326	111068	167652	55384
100	110002	164930	54928	76187	114165	37978	110015	163655	53640	109963	166000	54837

表一為各個演算法在使用 Random reference string 時的表現情形，接下來我們會透過更細項的圖表做進一步討論與分析。

2.1.1 Page fault 與 frame 數量的關係



圖一：各演算法使用 Random reference string 時，在不同 frame 數量情形下的 page fault 發生情形

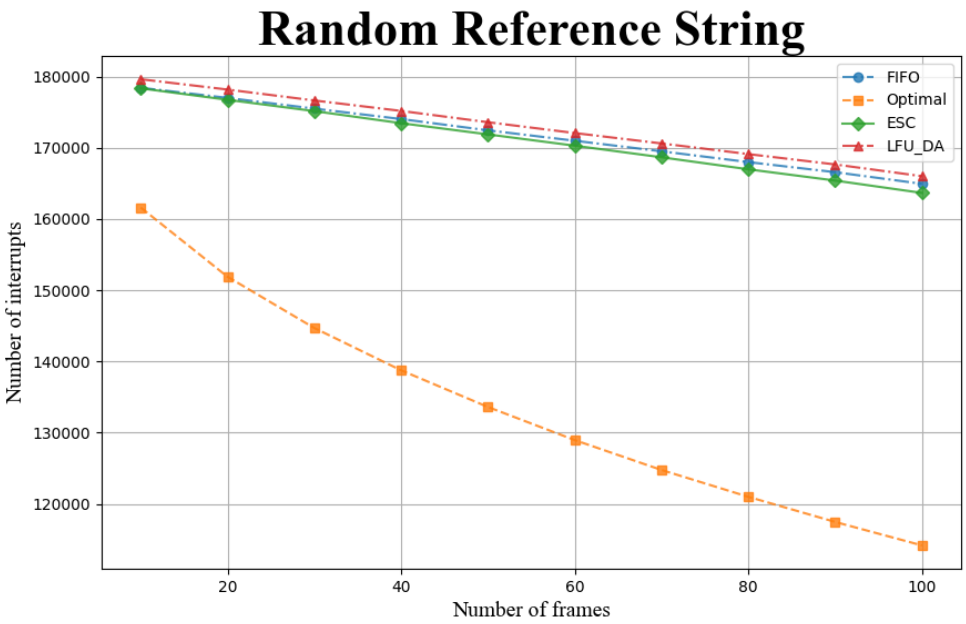
表二：各演算法使用 Random reference string 時，在不同 frame 數量情形下的 page fault 發生情形

Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	118991	107766	119013	118989
20	118046	101307	118017	118028
30	117055	96497	117056	117023
40	116074	92535	116015	116051
50	115029	89101	115048	115020
60	114052	86025	114063	114009
70	113054	83244	113082	113030
80	112049	80719	112050	112030
90	111103	78383	111063	111068
100	110002	76187	110015	109963

從圖一可以清楚地看到，Optimal 演算法的 page fault 數量遠低於其他演算法，而其他演算法的趨勢非常接近，彼此之間的差異不大。

由表二可看出， FIFO 演算法的 page fault 數量最多，其次為 ESC，而 LFU-DA 的表現略好於 ESC。最少的 page fault 發生數則是 Optimal 演算法。

2.1.2 Interrupt 數量與 frame 數量的關係



圖二：各演算法使用 Random reference string 時，在不同 frame 數量情形下的 interrupt 發生情形

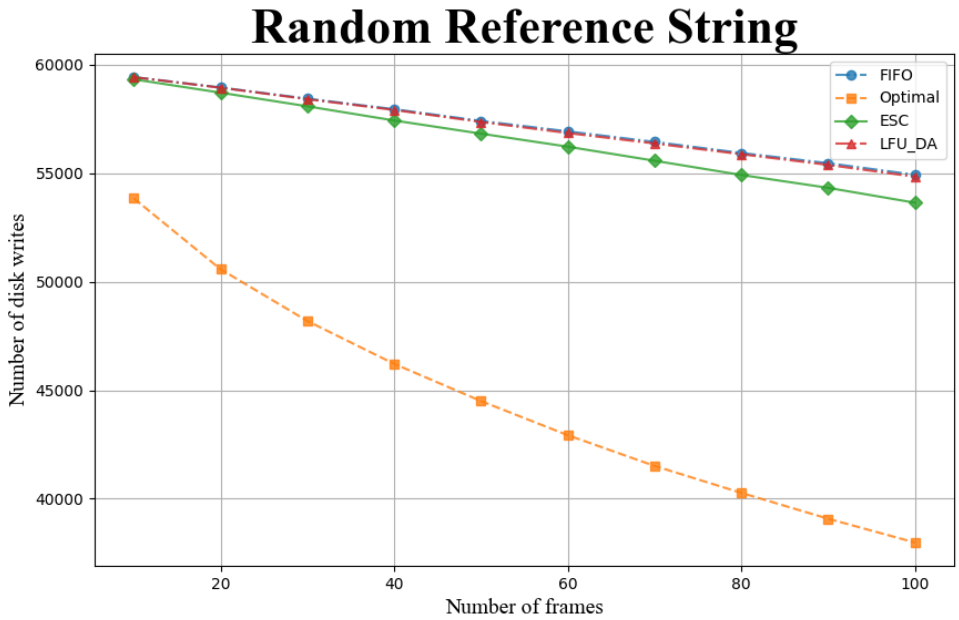
表三：各演算法使用 Random reference string 時，在不同 frame 數量情形下的 interrupt 發生情形

Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	178422	161610	178341	179617
20	176997	151883	176729	178168
30	175492	144688	175140	176635
40	174025	138750	173445	175167
50	172440	133601	171872	173582
60	170980	128952	170286	172061
70	169497	124754	168657	170600
80	167981	120986	166969	169105
90	166565	117458	165389	167652
100	164930	114165	163655	166000

圖二中，我們可以觀察到 Optimal 演算法的 interrupt 數量依然遠低於其他演算法。然而，在這個實驗中， LFU-DA 的 interrupt 數量是最高的，其次是 FIFO，再來是 ESC 與 Optimal。而在表三中也可以很清楚地觀察到這個趨勢。

LFU-DA 會有相對較高的 interrupt 數，其原因在於該演算法需要仰賴 timer 定期觸發 interrupt，使作業系統能夠定期衰減各個 page 在 counter 中的計數。因此，在比較 interrupt 時，LFU-DA 在這方面表現較為劣勢。

2.1.3 Disk writes 數量與 frame 數量的關係



圖三：各演算法使用 Random reference string 時，在不同 frame 數量情形下的 disk writes 發生情形

表四：各演算法使用 Random reference string 時，在不同 frame 數量情形下的 disk writes 發生情形

Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	59431	53844	59328	59428
20	58951	50576	58712	58940
30	58437	48191	58084	58412
40	57951	46215	57430	57916
50	57411	44500	56824	57362
60	56928	42927	56223	56852
70	56443	41510	55575	56370
80	55932	40267	54919	55875
90	55462	39075	54326	55384
100	54928	37978	53640	54837

由圖三與表四中，我們可以觀察到 disk writes 發生次數最多的為 FIFO，其次是 LFU-DA，ESC 的表現略好於 LFU-DA，而 Optimal 則有最少的 disk writes。

由於 ESC 與 LFU-DA 皆考慮了 dirty bit，因此在 disk writes 的次數上，這兩種演算法的表現都優於 FIFO。

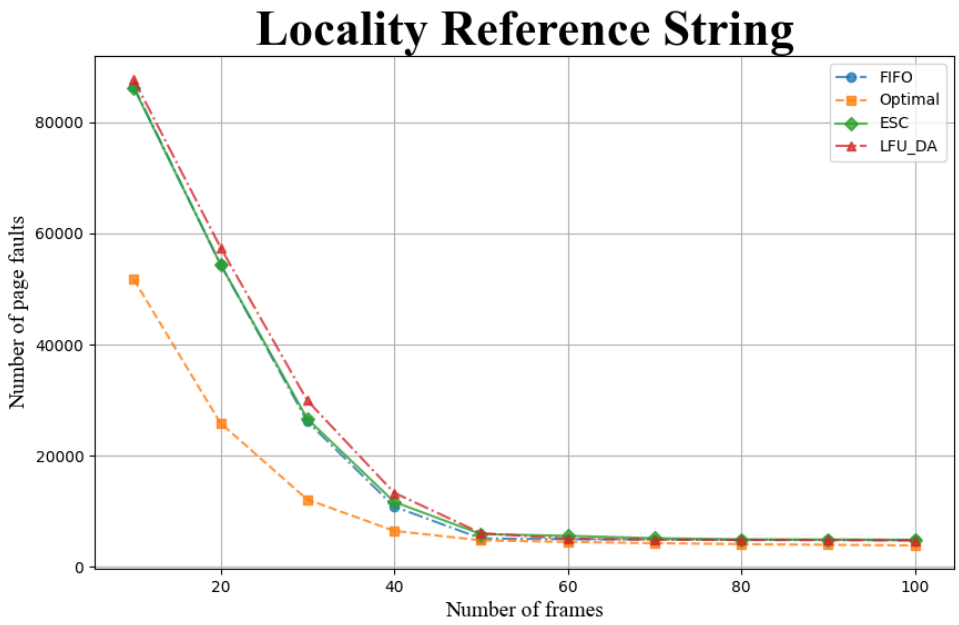
2.2 在 Locality reference string 中比較

表五：使用 Locality reference string 時各個演算法的表現

Frame Count	FIFO			Optiaml			ESC			LFU-DA		
	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes
10	86457	129435	42978	51752	77366	25614	86241	125743	39502	87764	131578	42614
20	54432	81460	27028	25846	38531	12685	54419	75465	21046	57486	85288	26602
30	26145	39012	12867	12092	18019	5927	26634	35016	8382	29916	44063	12947
40	10819	16125	5306	6454	9595	3141	11675	15067	3392	13316	19931	5415
50	5084	7541	2457	4744	7021	2277	5870	8245	2375	6046	10012	2766
60	4945	7326	2381	4457	6581	2124	5557	7874	2317	5038	8637	2399
70	4902	7257	2355	4240	6259	2019	5131	7422	2291	4865	8395	2330
80	4842	7157	2315	4059	5979	1920	4914	7154	2240	4812	8307	2295
90	4816	7127	2311	3940	5796	1856	4898	7096	2198	4792	8282	2290
100	4757	7030	2273	3830	5631	1801	4847	7012	2165	4727	8173	2246

表五為各個演算法在使用 Locality reference string 時的表現情況。接下來，我們將會透過更細項的圖表做進一步討論與分析。

2.2.1 Page fault 與 frame 數量的關係



圖四：各演算法使用 Locality reference string 時，在不同 frame 數量情形下的 page fault 發生情形

表六：各演算法使用 Locality reference string 時，在不同 frame 數量情形下的 page fault 發生情形

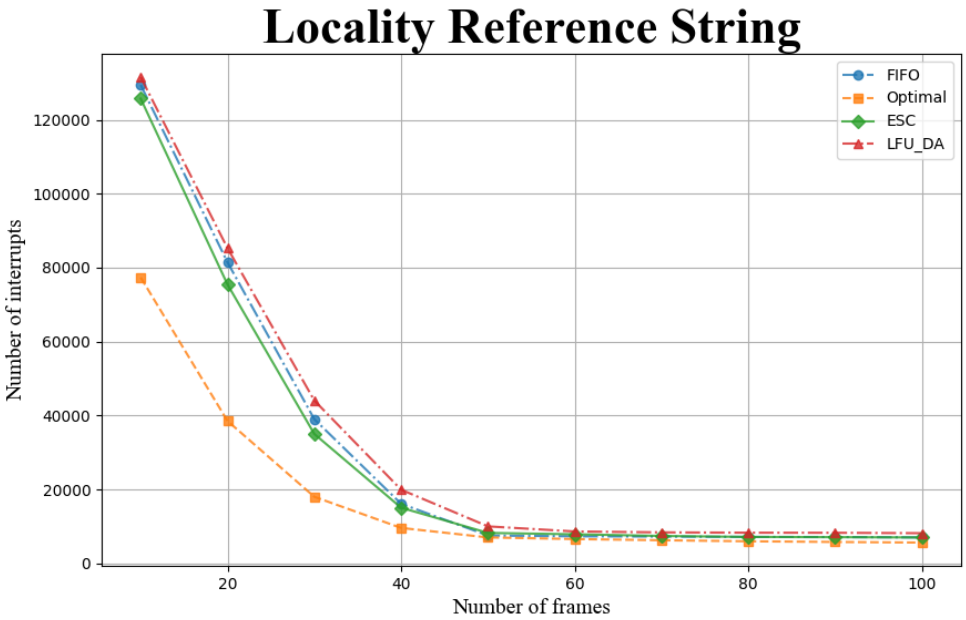
Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	86457	51752	86241	87764
20	54432	25846	54419	57486
30	26145	12092	26634	29916
40	10819	6454	11675	13316
50	5084	4744	5870	6046
60	4945	4457	5557	5038
70	4902	4240	5131	4865
80	4842	4059	4914	4812
90	4816	3940	4898	4792
100	4757	3830	4847	4727

由圖四與表六可以觀察到，Optimal 演算法在所有的 frame 數量中依然有著最佳的表現。而其他演算法則在不同 frame 數量下有著不同的表現。我們可以看到，當 frame 數量小於 50 時，LFU-DA 的 page fault 數量略高於其他演算法；但當 frame 超過 50 後，ESC 演算法的 page fault 數量開始高於其他演算法。

雖然我們設計的 LFU-DA 在較少 frame 的情況下表現不如預期，但在 frame 數增加時，其表現僅次於 Optimal。

在圖四中，我們也可以發現，使用 Locality reference string 時，由於 locality 特性的影響，FIFO、ESC 和 LFU-DA 的表現隨著 frame 數量增加逐漸接近 Optimal 的表現。

2.2.2 Interrupt 數量與 frame 數量的關係



圖五：各演算法使用 Locality reference string 時，在不同 frame 數量情形下的 interrupt 發生情形

表七：各演算法使用 Locality reference string 時，在不同 frame 數量情形下的 interrupt 發生情形

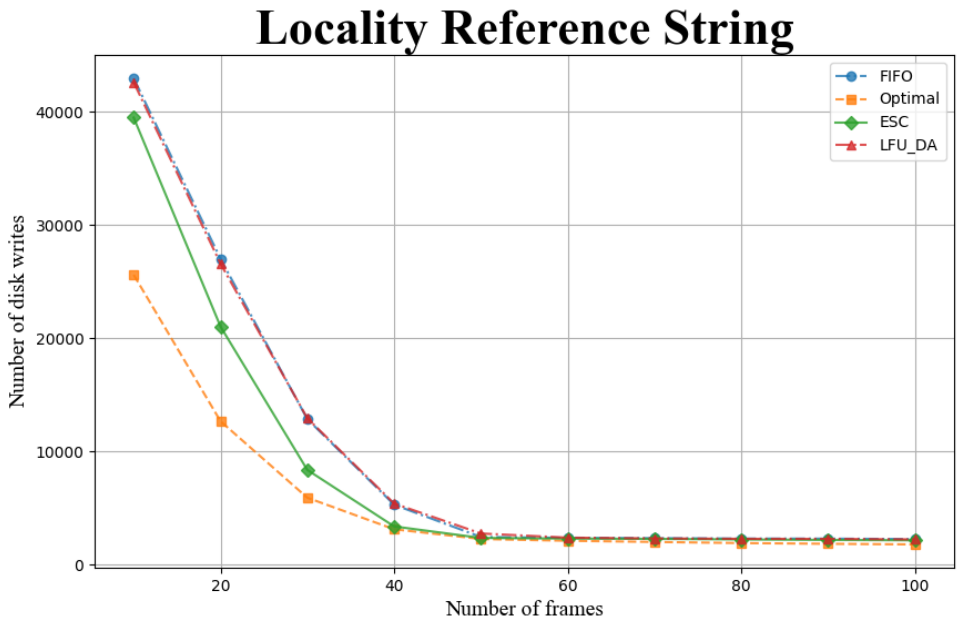
Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	129435	77366	125743	131578
20	81460	38531	75465	85288
30	39012	18019	35016	44063
40	16125	9595	15067	19931
50	7541	7021	8245	10012
60	7326	6581	7874	8637
70	7257	6259	7422	8395
80	7157	5979	7154	8307
90	7127	5796	7096	8282
100	7030	5631	7012	8173

圖五中，我們可以觀察 LFU-DA 的 interrupt 數量是最高的，其次是 FIFO，再來是 ESC 與 Optimal。而在表七中也可以很清楚地觀察到這個趨勢。

如同前一節所提到的，LFU-DA 的 interrupt 數量較高，主要是因為該演算法依賴 timer 定期觸發 interrupt，從而使作業系統能夠定期衰減各個 page 的 counter

計數。因此，在比較 interrupt 數量時，LFU-DA 的表現相對較弱。

2.2.3 Disk writes 數量與 frame 數量的關係



圖六：各演算法使用 Locality reference string 時，在不同 frame 數量情形下的 disk writes 發生情形

表八：各演算法使用 Locality reference string 時，在不同 frame 數量情形下的 disk writes 發生情形

Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	42978	25614	39502	42614
20	27028	12685	21046	26602
30	12867	5927	8382	12947
40	5306	3141	3392	5415
50	2457	2277	2375	2766
60	2381	2124	2317	2399
70	2355	2019	2291	2330
80	2315	1920	2240	2295
90	2311	1856	2198	2290
100	2273	1801	2165	2246

由圖六與表八中，我們可以觀察到 disk writes 發生次數最多的為 FIFO，其次是 LFU-DA，接著為 ESC，而 Optimal 則有最少的 disk writes。

這個一個結果也與前一節的分析一致，由於 ESC 與 LFU-DA 都考慮了 dirty

bit，因此在 disk writes 次數上，這兩種演算法的表現都優於 FIFO。

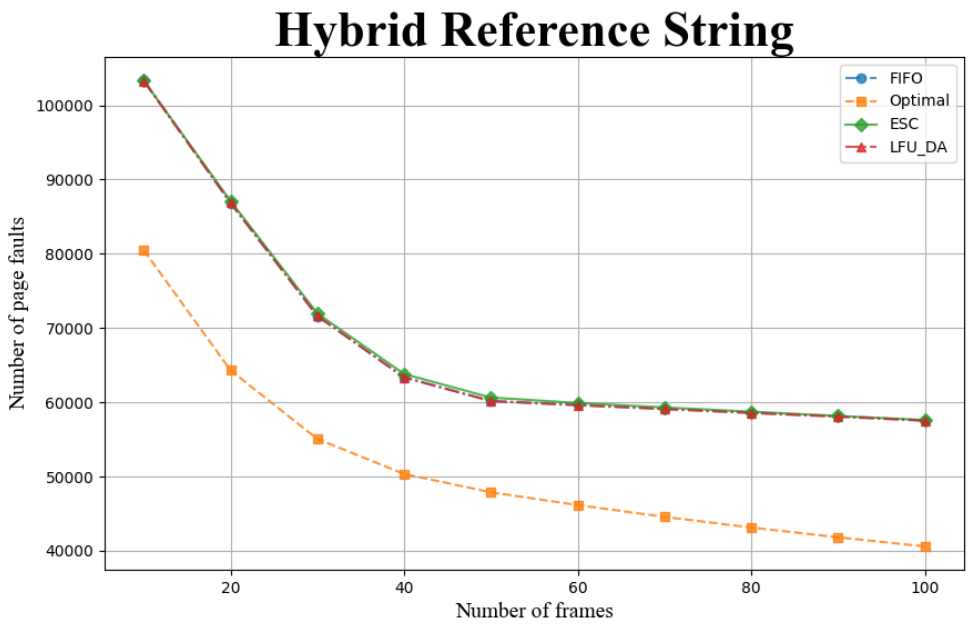
2.3 在 Hybrid reference string 中比較

表九：使用 Hybrid reference string 時各個演算法的表現

Frame Count	FIFO			Optiaml			ESC			LFU-DA		
	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes	Page Faults	Interrupts	Disk Writes
10	103365	154925	51560	80488	120614	40126	103329	153153	49824	103278	155406	50928
20	86838	130147	43309	64311	96322	32011	87102	127279	40177	86864	130303	42239
30	71594	107254	35660	55064	82448	27384	71956	105127	33171	71638	107636	34798
40	63357	94902	31545	50345	75354	25009	63804	94137	30333	63346	95575	31029
50	60185	90134	29949	47873	71665	23792	60630	90229	29599	60152	91244	29892
60	59643	89323	29680	46156	69110	22954	59905	89185	29280	59594	90400	29606
70	59129	88560	29431	44561	66699	22138	59297	88303	29006	59073	89609	29336
80	58595	87762	29167	43128	64501	21373	58727	87395	28668	58547	88818	29071
90	58110	87049	28939	41811	62521	20710	58152	86485	28333	58051	88078	28827
100	57549	86229	28680	40584	60683	20099	57607	85597	27990	57500	87274	28574

表九為各個演算法在使用 Hybrid reference string 時的表現情況。接著，我們將會透過更細項的圖表做進一步討論與分析。

2.3.1 Page fault 與 frame 數量的關係



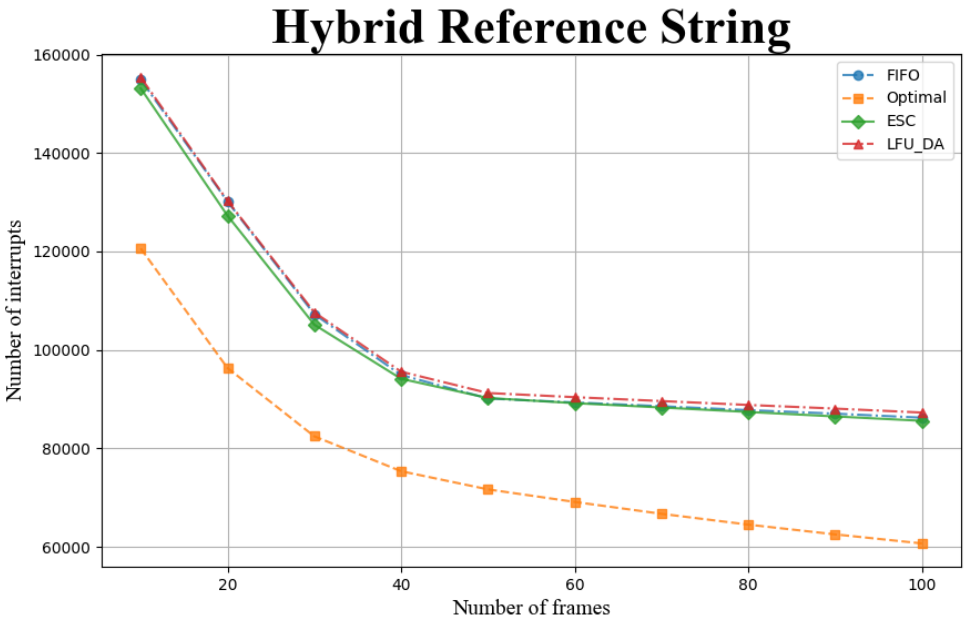
圖七：各演算法使用 Hybrid reference string 時，在不同 frame 數量情形下的 page fault 發生情形

表十：各演算法使用 Hybrid reference string 時，在不同 frame 數量情形下的 page fault 發生情形

Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	103365	80488	103329	103278
20	86838	64311	87102	86864
30	71594	55064	71956	71638
40	63357	50345	63804	63346
50	60185	47873	60630	60152
60	59643	46156	59905	59594
70	59129	44561	59297	59073
80	58595	43128	58727	58547
90	58110	41811	58152	58051
100	57549	40584	57607	57500

由圖七與表十可見，在大多數情況下，Optimal 演算法的依然表現最佳，而 LFU-DA 則是僅次於 Optimal。相較之下，ESC 在使用 Hybrid reference string 的情形下表現最不理想，FIFO 的表現則是略優於 ESC。

2.3.2 Interrupt 數量與 frame 數量的關係.



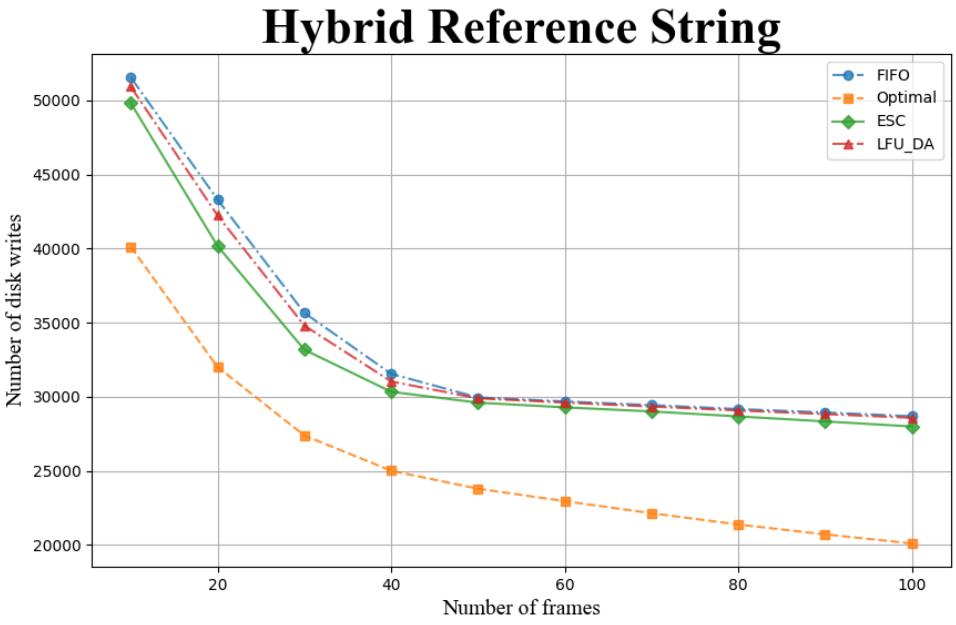
圖八：各演算法使用 Hybrid reference string 時，在不同 frame 數量情形下的 interrupt 發生情形

表十一：各演算法使用 Hybrid reference string 時，在不同 frame 數量情形下的 interrupt 發生情形

Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	154925	120614	153153	155406
20	130147	96322	127279	130303
30	107254	82448	105127	107636
40	94902	75354	94137	95575
50	90134	71665	90229	91244
60	89323	69110	89185	90400
70	88560	66699	88303	89609
80	87762	64501	87395	88818
90	87049	62521	86485	88078
100	86229	60683	85597	87274

圖八與表十一中，我們可以觀察 LFU-DA 的 interrupt 數量依然最高，其次是 FIFO，再來是 ESC 和 Optimal。這與前幾節的分析結果一致。LFU-DA 的 interrupt 數量較高，主要原因是該演算法依賴 timer 定期觸發 interrupt，使作業系統能夠定期衰減各個 page 的 counter 計數。因此，在比較 interrupt 數量時，LFU-DA 的表現相對較為劣勢。

2.3.3 Disk writes 數量與 frame 數量的關係



圖九：各演算法使用 Hybrid reference string 時，在不同 frame 數量情形下的 disk writes 發生情形

表十二：各演算法使用 Hybrid reference string 時，在不同 frame 數量情形下的 disk writes 發生情形

Frame Count	FIFO	Optiaml	ESC	LFU-DA
10	51560	40126	49824	50928
20	43309	32011	40177	42239
30	35660	27384	33171	34798
40	31545	25009	30333	31029
50	29949	23792	29599	29892
60	29680	22954	29280	29606
70	29431	22138	29006	29336
80	29167	21373	28668	29071
90	28939	20710	28333	28827
100	28680	20099	27990	28574

由圖九與表十二中可以觀察到，disk writes 發生次數最多的是 FIFO，其次是 LFU-DA，接著是 ESC，而最少的是 Optimal。這一結果與前幾節的分析相符。由於 ESC 和 LFU-DA 都考慮了 dirty bit，因此在 disk writes 次數上，這兩種演算法的表現優於 FIFO。

3. 結論

總結實驗結果，Optimal 演算法在所有情境中表現最佳，無論是 page fault 還是 disk writes 的次數都遠低於其他演算法。而我設計的 LFU-DA 演算法表現次於 Optimal，特別是在 frame 數量較大時可以有更優越的效能。然而，由於其依賴定期的 timer 觸發以衰減 counter，導致 interrupt 數量相對較高，這是其劣勢之一。

ESC 演算法的表現介於 LFU-DA 和 FIFO 之間，雖然其在 disk writes 次數上表現優異，但 page fault 數量在某些情況下相對較高。FIFO 演算法則在所有測試中表現最差，無論是 page fault 還是 disk writes 數量都高於其他演算法。

總結來看，LFU-DA 儘管在 interrupt 數量上有一定劣勢，但其在 page fault 和 disk writes 的控制上表現優異。未來可以針對其 interrupt 數量進行改善，以進一步提升性能。