The University of Adelaide

OS Report

Le Thuy An Phan – a1874923 – Equal
Tri Hai Huynh – a1906375 – Equal

Introduction

Efficient management of main memory is a central challenge for operating systems. Virtual memory addresses this by giving programs the illusion of a larger memory than physically available: data not currently in RAM can be stored on secondary storage (disk) and brought into memory on demand. Virtual memory is divided into fixed-size units called *pages*, which are loaded into a limited number of physical page frames in RAM. When a program accesses a page that is not resident in memory, a *page fault* occurs and the operating system must choose which resident page to evict to make room for the requested page. The choice of page-replacement policy directly affects the page-fault rate and therefore overall system performance, because disk I/O is orders of magnitude slower than memory access. Inefficient policies produce high fault rates and degrade throughput and responsiveness. In this project we investigate four page-replacement policies implemented in our simulator: `Rand` (Random), `LRU` (Least Recently Used), `Clock` (approximation to LRU), and `Optimal` (theoretical benchmark). The purpose of this report is to evaluate and compare the performance of these algorithms on real-world memory traces. More specifically, we aim to answer the following research questions:

1. How does the number of available page frames affect the page-fault rate for each algorithm in a set of population of types of memory trace?
2. Which algorithm performs best when the number of available frames is insufficient (i.e., extreme thrashing)?
3. Do results vary significantly across different memory access patterns (different traces)?

Methodology

This section describes the simulation environment, and the experimental design used to collect performance data.

Simulator and environment setup

We developed a simulator in Python that models a single-level page table and evaluates page-replacement policies on real trace inputs. To ensure fair comparison the simulator used the following fixed parameters:

- **Page and frame size:** 4 KB.
- **Trace inputs:** Each run processes one memory trace containing 1 000 000 memory accesses. The traces used in this report are `bzip`, `gcc`, `sixpack` and `swim`.
- **Primary metric:** *Page-fault rate* $= \frac{\text{Total Page Faults}}{\text{Total Memory Accesses}}$. Total disk reads and writes are also recorded, but the page-fault rate is used as the core metric for visualisation and analysis because it directly reflects the frequency of expensive disk I/O events.

Experimental design

- **Low Frame Size:** evaluate robustness and thrashing behaviour when frames are scarce.
- **Medium Frame Size:** identify where the largest marginal improvement occurs and estimate the program's working-set size.
- **High Frame Size:** show convergence of page-fault rates as additional frames yield diminishing returns.
- **Statistical Inference:** to determine if the observed differences between the algorithms were statistically significant, a Matched Pair t-test was employed, however with the assumption that the mean differences between the algorithms are normally distributed.
- **Mean Table:** a mean table of different segments of a specific trace of each algorithm to numerically describe the overall performance of each algorithm.
- **Mean Table:** each result of a trace file is plotted by Matplotlib with x-axis as frame size and y-axis as page fault rate in a line graph.

GCC Trace

Segment	LRU	RAND	CLOCK	OPTIMAL
1–200	0.0644	0.0806	0.0678	0.0431
1–10	0.2744	0.3250	0.2957	0.2228
10–75	0.0805	0.1023	0.0838	0.0534
75–200	0.0399	0.0507	0.0420	0.0239
Mean	0.1148	0.1397	0.1223	0.0858

Table 1: Mean fault rates by segment and algorithm

OPTIMAL consistently outperforms others with the overall fault rate mean of 0.0858, while RAND is the worst with fault rate of 0.1397. LRU and CLOCK are close, with LRU slightly better in thrashing scenario, low frames (1-10: $0.2744 < 0.2957$), suggesting LRU performs better than CLOCK's approximation of LRU in high-fault scenarios. The overall performance is on average $+0.02 \rightarrow +0.05$ fault rate from the OPTIMAL.

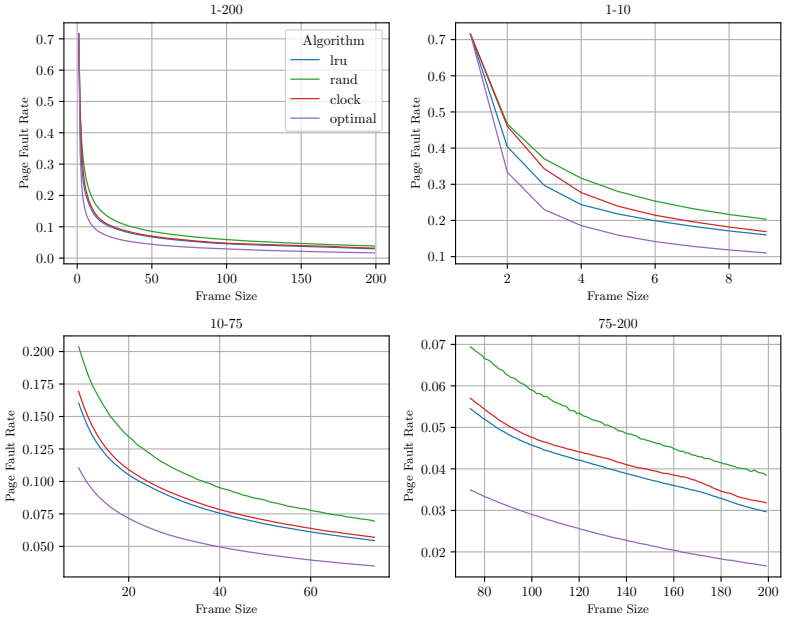
Comparison	n	t_stat	df	p_value	mean_diff	diff_low	diff_high
`lru_vs_rand`	4	−2.8009	3	0.0678	−0.0249	−0.0531	0.0034
`lru_vs_clock`	4	−1.6355	3	0.2005	−0.0075	−0.0222	0.0071
`lru_vs_optimal`	4	3.6863	3	0.0346	0.0290	0.0040	0.0540
`rand_vs_clock`	4	3.8768	3	0.0304	0.0173	0.0031	0.0315
`rand_vs_optimal`	4	3.2176	3	0.0487	0.0538	0.0006	0.1071
`clock_vs_optimal`	4	2.9497	3	0.0600	0.0365	−0.0029	0.0759

Table 2: Pairwise comparison statistics[3]

Based on the samples, I am 95% confident that the following differences are statistically significant at the p-value < 0.05[3]:

- Mean fault rate difference (LRU − OPTIMAL) is between [0.0040, 0.0540]
- Mean fault rate difference (RAND − CLOCK) is between [0.0031, 0.0315]
- Mean fault rate difference (RAND − OPTIMAL) is between [0.0006, 0.1071]

Thus, at the 95% confidence level we conclude that LRU performs significantly worse than OPTIMAL, and RAND performs significantly worse than CLOCK and OPTIMAL on the GCC memory trace sample.



Across all segments, fault rates display exponential decay as frame size increases. RAND has the highest rates, visually separating from others, supporting its significant inferiority to CLOCK and OPTIMAL. LRU and CLOCK lines stay closely with each other in all segments, explaining the high p_value ≈ 0.2005. In extreme thrashing (1-10), LRU edges slightly lower, but all go to convergence by 75-200 aligns with the overall performance of the Table 1. Overall, the visuals reinforce the theoretical sample trends of GCC memory trace. Additionally, the data here has a strong evidence of supporting each other.

Bzip Trace

Segment	LRU	RAND	CLOCK	OPTIMAL
1–200	0.0072	0.0089	0.0083	0.0063
1–10	0.1218	0.1503	0.1429	0.1123
10–75	0.0022	0.0028	0.0025	0.0014
75–200	0.0007	0.0009	0.0008	0.0005
Mean	0.0330	0.0407	0.0386	0.0301

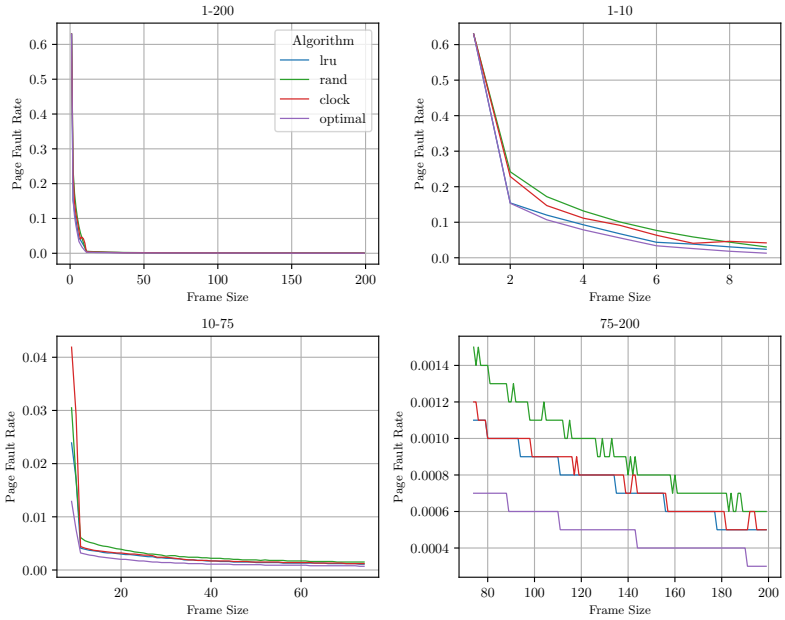
Table 3: Mean fault rates by segment and algorithm

OPTIMAL has the lowest overall mean fault rate (0.0301) while RAND is highest (0.0407). CLOCK and RAND are very close in every segment, and the mean gap $0.0056_{clock-lru} > 0.0021_{rand-clock}$, indicating CLOCK is closer to RAND than to LRU despite being an approximate of LRU. During the working-set frame count (10-75) the difference starts to normalise. At higher frame counts (75-200), LRU approaches OPTIMAL, while CLOCK remains to be closer to RAND than LRU.

Comparison	n	t_stat	df	p_value	diff_low	diff_high
`lru_vs_rand`	4	−1.1193	3	0.3445	−0.0298	0.0143
`lru_vs_clock`	4	−1.0961	3	0.3531	−0.0221	0.0108
`lru_vs_optimal`	4	1.2826	3	0.2898	−0.0042	0.0099
`rand_vs_clock`	4	1.1867	3	0.3208	−0.0035	0.0077
`rand_vs_optimal`	4	1.1592	3	0.3303	−0.0185	0.0397
`clock_vs_optimal`	4	1.1526	3	0.3326	−0.0150	0.0320

Table 4: Pairwise comparison statistics[3]

Based on the samples, at 95% confidence level, none of the pairwise differences are statistically significant, all p-value > 0.05[3].
Thus, we cannot conclude any algorithm differs significantly from another on the Bzip memory trace sample. However, the statistics show that the weakest difference is between LRU-CLOCK which contradicts with the mean table 3 that shows otherwise.



Across all segments, fault rates display exponential decay as frame size increases. In the frame size of (20-200), RAND and OPTIMAL serve as the upper and lower bounds. However, the first segment (1-200) display a local lump from CLOCK, can be explained by CLOCK temporarily degenerate to FIFO[2]. As we also know that FIFO will suffer from the Belady's Anomaly. So, CLOCK exhibit the anomaly, in that when increasing frame size the fault rate goes up[1]. Overall, the visuals reinforce theoretical assumption of the table 3 as CLOCK-RAND go closely across. While, after the anomaly, CLOCK and LRU consistently approache OPTIMAL at larger frame size, which supports inference from table 4.

Swim Trace

Segment	LRU	RAND	CLOCK	OPTIMAL
1–200	0.0454	0.0594	0.0490	0.0288
1–10	0.3723	0.4233	0.3935	0.2843
10–75	0.0615	0.0859	0.0666	0.0336
75–200	0.0125	0.0182	0.0138	0.0068
Mean	0.1229	0.1467	0.1307	0.0884

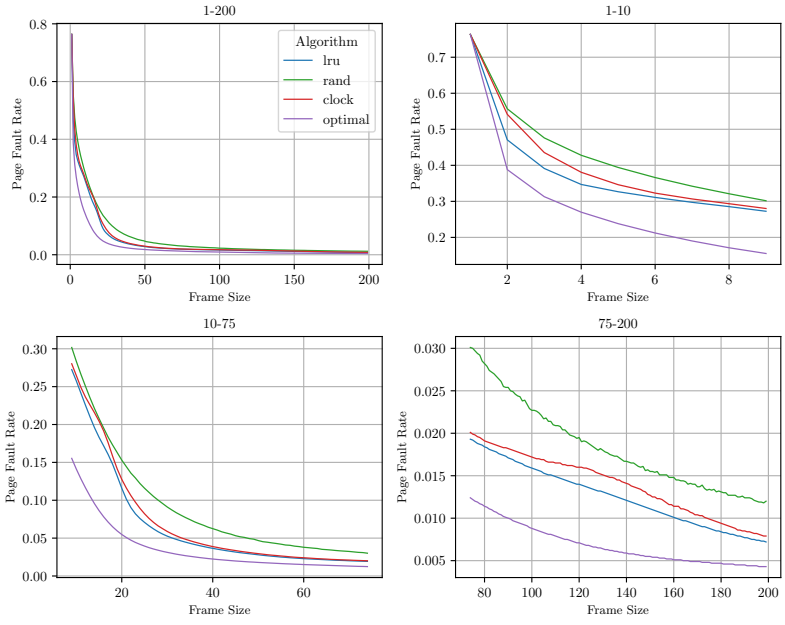
Table 5: Mean fault rates by segment and algorithm

OPTIMAL consistently outperforms others with the overall mean 0.0884, while RAND is worst (0.1467). LRU and CLOCK are close, with LRU slightly better in thrashing scenario, low frames (1-10: $0.3723 < 0.3935$), suggesting LRU performs better than CLOCK's approximation of LRU in high-fault scenarios. The overall performance is on average $+0.04 \rightarrow +0.06$ fault rate from the OPTIMAL.

Comparison	n	t_stat	df	p_value	mean_diff	diff_low	diff_high
`lru_vs_rand`	4	−2.4141	3	0.0947	−0.0238	−0.0551	0.0076
`lru_vs_clock`	4	−1.7201	3	0.1839	−0.0078	−0.0222	0.0066
`lru_vs_optimal`	4	1.8794	3	0.1568	0.0346	−0.0240	0.0931
`rand_vs_clock`	4	2.8877	3	0.0631	0.0160	−0.0016	0.0336
`rand_vs_optimal`	4	2.0712	3	0.1301	0.0583	−0.0313	0.1479
`clock_vs_optimal`	4	1.8488	3	0.1616	0.0424	−0.0305	0.1152

Table 6: Pairwise comparison statistics[3]

Based on the samples, at 95% confidence level, none of the pairwise differences are statistically significant, all p-value > 0.05[3].
Thus, we cannot conclude any algorithm differs significantly from another on the Swim memory trace sample.



Across all segments, fault rates display exponential decay as frame size increases. RAND and OPTIMAL serve as a clear uppper and lower bounds. The relationship between RAND-LRU-CLOCK in the visuals support the inference from Table 6, as the p-value of LRU-CLOCK is highest therefore go together closely while having insignificant difference with RAND. In the working set frame size (10-75), with the current evidence and visuals CLOCK has a small lump this shows that CLOCK has almost degenerated to FIFO[2]. Overall, the visuals reinforce the theoretical sample trends from Table 5, which LRU-CLOCK on average go closely together and RAND is the worst on average, and Table 6 of Swim trace. Additionally, the data here has a strong evidence of supporting each other.

Sixpack Trace

Segment	LRU	RAND	CLOCK	OPTIMAL
1–200	0.0488	0.0630	0.0523	0.0317
1–10	0.3054	0.3595	0.3315	0.2401
10–75	0.0656	0.0848	0.0688	0.0392
75–200	0.0204	0.0290	0.0222	0.0116
Mean	0.1101	0.1341	0.1187	0.0807

Table 7: Mean fault rates by segment and algorithm

OPTIMAL consistently outperforms others with the overall fault rate mean of 0.0807, while RAND is worst with fault rate at 0.1341. LRU and CLOCK are close, with LRU slightly better in thrashing scenario, low frames (1-10: $0.3054 < 0.3315$), suggesting LRU performs better than CLOCK's approximation of LRU in high-fault scenarios. The overall performance is on average $+0.03 \rightarrow +0.05$ fault rate from the OPTIMAL.

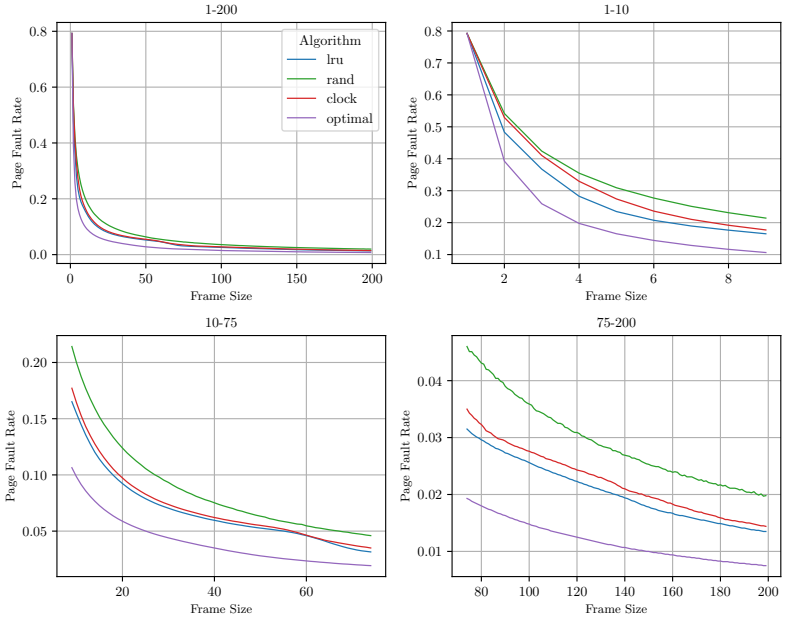
Comparison	n	t_stat	df	p_value	mean_diff	diff_low	diff_high
`lru_vs_rand`	4	−2.3425	3	0.1010	−0.0240	−0.0567	0.0086
`lru_vs_clock`	4	−1.4841	3	0.2344	−0.0087	−0.0272	0.0099
`lru_vs_optimal`	4	2.3530	3	0.1000	0.0294	−0.0104	0.0692
`rand_vs_clock`	4	3.3342	3	0.0446	0.0154	0.0007	0.0301
`rand_vs_optimal`	4	2.3502	3	0.1003	0.0534	−0.0189	0.1258
`clock_vs_optimal`	4	2.0905	3	0.1277	0.0380	−0.0199	0.0960

Table 8: Pairwise comparison statistics[3]

Based on the samples, I am 95% confident that the following differences are statistically significant at the p-value < 0.05[3]:

- Mean fault rate difference (RAND − CLOCK) is between [0.0007, 0.0301]

Thus, at the 95% confidence level we conclude that RAND performs significantly worse than CLOCK on the Sixpack memory trace.



Across all segments, fault rates display exponential decay as frame size increases. RAND and OPTIMAL serve as the upper and lower bounds . LRU and CLOCK lines stay closely with each other in all segments, explaining the high p_value ≈ 0.2344 in Table 8. In extreme thrashing (1-10), LRU edges slightly lower, but all go to convergence by 75-200 aligns with the overall performance of the Table 7. Overall, the visuals reinforce the theoretical sample trends of Sixpack memory trace. Additionally, the data here has a strong evidence of supporting each other.

Conclusion

In conclusion, the evidence strongly supports that the number of available page frames affect the page-fault rate for each algorithm across different types of memory traces. For example, when frame size increase, page-fault rate usually drop, but it depend on the algorithm, such as CLOCK might degenerate to FIFO which would produce more faults in sequential traces, while LRU behaves better. There exists a working set size that is the optimal amount of frames which yield a good page fault rate. Regarding of which algorithm performs best in extreme thrashing, the evidence shows LRU is consistent through memory traces under such condition. Finally, the evidence supports that the result does not have big variation between traces, by that LRU stays as a top performer, and CLOCK often approximates LRU correctly, even in phased traces like bzip where bump happen but not too disruptive overall. This suggests algorithms are robust in general.

References

[1] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. Operating systems: Three easy pieces, 2014.

[2] Cornell University. Lecture 15: More on page replacement.

[3] Eberly College of Science, The Pennsylvania State University. 10.3 - paired t-test.