Project 4
Memory management for Swapping and Paging

Dhruvi Kothari

Inderpreet Kaur

Max Khan

Andrew Li

Introduction

This project aims to simulate memory management using various page replacement strategies. Processes are designed to mimic real-world memory access behavior, specifically leveraging the locality of reference principle. Each process begins execution from page 0 and references a page from its virtual address space every 100 milliseconds. The locality of reference principle means that after accessing page 'i', there's a 70% chance the next access will be to page 'i', 'i-1', or 'i+1' (with page numbers wrapping around). In other cases, the absolute value of Δi will be greater than 1.

Workload Generation

A total of 150 jobs were generated with attributes such as page size, arrival time, and service duration. Job sizes were randomly selected from 5, 11, 17, or 31 pages, and durations ranged from 1 to 5 seconds. Jobs were sorted based on arrival time and stored in a linked list to simulate realistic workload scheduling.

Simulator

The simulator initializes memory with 100 pages and sequentially processes tasks. It starts a task only if there are at least 4 free pages available. Each memory access is logged and evaluated for a hit or miss. When a page miss occurs and no free pages are available, the appropriate page replacement algorithm is invoked to evict a page.

The simulator performs 5 runs for each algorithm, each lasting 1 minute. Metrics such as hit ratio, miss ratio, and the number of processes swapped into memory are recorded at the end of each run and averaged for reporting.

Page Replacement Strategies:

First-In-First-Out (FIFO)

FIFO removes the oldest page in memory, following a simple queue logic. While easy to implement, it may evict pages that are still actively used, leading to suboptimal performance in workloads with temporal locality.

Least Recently Used (LRU)
LRU evicts the page that has not been used for the longest time. It assumes that pages used recently are more likely to be used again soon. Though more effective than FIFO, it requires tracking the order of access.

Least Frequently Used (LFU)
LFU targets pages that have been accessed the least number of times. The idea is to retain pages that are regularly used. It performs well in stable access patterns but can lag behind in changing workloads unless usage history is periodically reset.

Most Frequently Used (MFU)
MFU does the opposite of LFU — it evicts pages with the highest frequency. The assumption is that such pages have completed their use and are less likely to be accessed again soon. This strategy works well in limited scenarios.

Random Replacement
Random replacement selects a page to evict entirely at random. It is the easiest to implement and has minimal overhead, but generally performs worse than algorithms that consider access patterns.

Output

Algorithm	Hit Ratio (%)	Miss Ratio (%)	Processes started
FIFO	66.51	33.49	150
LRU	69.51	30.49	150
LFU	65.37	34.63	150
MFU	69.12	30.88	150
Random	67.24	32.76	150

Observation

The results were averaged over 5 simulation runs for each algorithm. LFU produced the highest hit ratio, indicating strong performance in retaining frequently accessed pages. FIFO and MFU showed slightly lower effectiveness due to their inflexibility in adapting to temporal changes in memory usage. Random performed the worst overall, as expected, since it does not incorporate any access history or patterns.