Advanced Operating Systems
Project-3 Group 2

Group Members:

1. Aifaz Maknojia
2. Samarth Kulkarni
3. Samuel Karumanchi
4. Sashidhar Kancharla
5. Vedant Raj Kavi

Introduction

This project investigates memory management in a multi-process system by simulating various page replacement algorithms. The aim is to observe how each algorithm manages memory through paging and swapping under high workload conditions. We developed a simulator in C that creates a process workload, manages it in memory, and applies one of five-page replacement methods: FIFO (First-In-First-Out), LRU (Least Recently Used), LFU (Least Frequently Used), MFU (Most Frequently Used), and Random Replacement. Running each algorithm on the same workload allows us to evaluate their performance by analysing page hits, misses, and successful swaps.

2. System Design and Implementation

Memory and Process Structure

- **Memory Layout**: The system's main memory is configured as 100 MB, with each page set to 1 MB, providing a total of 100 pages. Initially, all pages are free.
- **Free Page List**: Memory pages are organized as a linked list, allowing for easy management of available pages.
- **Process Workload**: The workload comprises 150 processes, each with random arrival times, memory sizes (5, 11, 17, or 31 MB), and durations (ranging from 1 to 5 seconds). These processes are queued and sorted by arrival time. A process starts only when there are at least 4 free pages available.
- **Locality of Reference**: Every 100 ms, each process references a new page, with a 70% likelihood of accessing the same page or a neighbouring page, simulating

"locality of reference." In other cases, a random page is selected to add an element of randomness, though it remains biased towards nearby pages.

Page Replacement Algorithms

To determine which page to evict when memory is full, the simulator uses five different strategies:

- **FIFO (First-In-First-Out)**: Removes the oldest page, favouring pages that have been in memory the longest.
- **LRU (Least Recently Used)**: Evicts the least recently accessed page, thereby keeping recently accessed pages for longer.
- **LFU (Least Frequently Used)**: Removes the page with the lowest access frequency, operating on the assumption that infrequently accessed pages are less likely to be needed soon.
- **MFU (Most Frequently Used)**: Opposite to LFU, this strategy evicts the most frequently accessed page, assuming high access frequency may imply that a fresh page will soon be required.
- **Random Replacement**: Selects a page to evict at random, without considering access history or frequency.

Data Collection for Each Run

For each simulation run, the following metrics were recorded:

- **Page Hits and Misses**: A page hit indicates that the page is already in memory, while a miss triggers a page load.
- **Memory Map**: A snapshot of memory, indicating which pages are currently occupied.
- **Replacement Logs**: Details on page replacements, including timestamps, process names, page references, and any evictions.

3. Execution and Observations

The simulator runs each of the five algorithms for a duration of one minute, with five runs in total. Every process references a page every 100 ms, and when memory is fully allocated, the chosen algorithm evicts a page to accommodate the new reference. The performance of each algorithm was evaluated by analyzing hit/miss ratios and the average number of processes swapped in over each run.

Results for each page replacement algorithm are summarized below:

Algorithm	**Page Hit Ratio**	**Page Miss Rate**	**Average Processes Swapped-in**
FIFO	79.73%	20.27%	1034
LFU	79.61%	20.39%	1039
LRU	79.80%	20.20%	1031
MFU	78.99%	21.01%	1066
RANDOM	79.90%	20.10%	1040

Individual Observations:

- **FIFO**: Achieved a 79.73% hit rate, with an average of 1034 processes swapped in. It efficiently managed memory by freeing the oldest pages.
- **LFU**: Had a hit rate of 79.61%, with 1039 processes swapped in, prioritizing the eviction of less frequently used pages.
- **LRU**: Demonstrated the highest hit rate at 79.80%, with an average of 1031 processes swapped in, effectively retaining recently accessed pages.
- **MFU**: Recorded the lowest hit rate at 78.99%, with 1066 processes swapped in, indicating that retaining frequently accessed pages wasn't the most effective approach.
- **Random Replacement**: Achieved a 79.90% hit rate with 1040 processes swapped in, but this strategy did not prioritize page retention.

5. Conclusion

This simulation illustrates the unique memory management approaches of each page replacement strategy. **LRU** performed the best, retaining pages that were frequently accessed. **FIFO** also achieved good results by following a straightforward approach of removing the oldest pages. **LFU** balanced between retaining frequently used pages and freeing up space effectively. **MFU** was less efficient, as holding frequently accessed pages didn't yield high performance. **Random Replacement** worked reasonably well for unpredictable access patterns but lacked the ability to prioritize important pages. Each algorithm's results emphasize the strengths and weaknesses of their respective strategies for handling memory in a multi-process environment.