Santa Clara University

Computer Science and Engineering

Advanced Operating System(CSEN 383)

Project 4

Group 8

Nehru Yalla

Anju Varghese

Jefferson Warie

Avani Vaidya

Swapping and Paging Simulation

Introduction

This C-language project delves into the intricate world of operating system memory management by simulating the concurrent execution of processes and their interaction with virtual memory. It focuses on the critical mechanism of paging, where a process's memory is divided into fixed-size pages, allowing for efficient allocation and utilization of system resources.

The simulation incorporates a variety of page replacement algorithms – FCFS, LRU, LFU, MFU, and Random – each representing a distinct strategy for deciding which page to evict from physical memory when a new page needs to be loaded. This provides a comparative framework for evaluating the effectiveness of these algorithms in different scenarios.

Locality of reference

Central to the simulation is the concept of locality of reference, a fundamental principle in computer science that states that a program tends to access the same set of memory locations repeatedly over a short period. The simulation models this behavior by assuming a high probability that a process's next memory reference will be to the current, previous, or next page.

The code itself is structured methodically, defining constants and data structures for pages, processes, and page lists, and setting parameters for various aspects of the simulation, such as hit/miss counters and the chosen page replacement algorithm. Multiple runs are executed in a loop, each simulating the arrival, execution, and page referencing of processes. Throughout this process, the chosen page replacement algorithm dynamically determines which pages to retain in memory and which to swap out to disk when memory pressure arises.

After referencing a page i, there is a 70% probability that the next reference will be to page i, i-1, or i+1. i wraps around from 10 to 0. In other words, there is a 70% probability that for a given i, Δi will be -1, 0, or +1. Otherwise,$|\Delta i| > 1$.

Code Description

The code is structured as follows:

- It begins by defining constants and data structures, including pages, processes, and a list of pages.

- It initializes various parameters such as paging options, hit and miss counters, and a function pointer for the selected page replacement algorithm.

- A loop simulates multiple runs of the system.

- Within each run, it initializes a list of pages and a queue of processes.

- Processes are created with random attributes like process ID, page count, arrival time, duration, and current page.

- Processes are sorted based on arrival time.

- The simulation then advances in time, loading processes into memory and simulating page references.

- Page replacement algorithms (e.g., FCFS, LRU, LFU, MFU, or Random) are invoked when needed.

- Hit and miss statistics are tracked for each run.

- At the end of each run, the average number of successfully swapped-in processes is calculated.

- The final hit-miss ratio is computed and displayed.

Code Explanation

The code consists of various functions, each responsible for a specific task. Here's a brief explanation of each of the page replacement algorithms used:

- FCFS_FUNCTION`: Implements the First-Come-First-Served (FCFS) page replacement algorithm, replacing the page with the earliest arrival time.

- LRU_FUNCTION`: Implements the Least Recently Used (LRU) page replacement algorithm, replacing the page that has not been used for the longest time.

- LFU_FUNCTION`: Implements the Least Frequently Used (LFU) page replacement algorithm, replacing the page with the fewest references.

- MFU_FUNCTION`: Implements the Most Frequently Used (MFU) page replacement algorithm, replacing the page with the most references.

- R_FUNCTION`: Implements a random page replacement algorithm, selecting a page to replace randomly.

Conclusion

This project not only simulates the mechanics of paging and page replacement but also sheds light on their real-world implications. By analyzing the hit/miss ratios and successful swap rates under different algorithms, one can gain a deeper understanding of how memory management choices can affect system performance. The insights gained from this simulation can be applied to optimize memory usage in various domains, from operating systems and databases to virtual machines and cloud computing environments.

Furthermore, the project serves as a stepping stone for further exploration into advanced memory management techniques, such as prefetching, working sets, and page coloring. It encourages experimentation and critical thinking about the trade-offs involved in different approaches to memory management.

Results:

Algorithm	**Hit Ratio**	**Pages Swapped**
FCFS	0.696496	1466
LFU	0.656127	1648
LRU	0.693575	1485
MFU	0.670170	1587
Random	0.694551	1517

The best performance in terms of both Hit Ratio and Pages Swapped - we got is First Come First Serve Algorithm