Cache Replacement Policy Simulator

Adhi Kesavan R[RA2411030050061]

San Clinton L[RA2411030050062]

Sasi Ganesh M R[RA2411030050063]

School of Computing, SRM Institute of Science & Technology, Tiruchirappalli, India
adhi.kesavan2024@srmist.edu.in, san.clinto2024@srmist.edu.in,
sasi.ganesh2024@srmist.edu.in

Abstract

This paper presents a Python-based simulator designed to evaluate and compare various cache replacement policies, including FIFO, LRU, LFU, and Clock algorithms. The simulator aims to provide an educational tool for students to understand cache behavior through experimentation, bridging the gap between theoretical learning and practical implementation. The tool supports multiple cache organizations such as direct-mapped, set-associative, and fully associative caches. Performance is measured in terms of hit/miss ratio and execution time under varying workloads.

Keywords— Cache replacement, LRU, LFU, FIFO, Clock, Cache Simulator, Operating Systems

I. Introduction

In modern computer systems, cache memory plays a crucial role in improving performance by storing frequently accessed data. However, cache capacity is limited, requiring efficient replacement policies to decide which cache lines to evict. Although several algorithms exist, such as FIFO, LRU, LFU, and Clock, most undergraduate students encounter them only in theory. This project introduces a Cache Replacement Policy Simulator that allows users to analyze these algorithms practically. The simulator integrates core concepts from Operating Systems, Computer Organization, and Data Structures.

II. Problem Statement

Cache performance depends heavily on the replacement policy used. While FIFO, LRU, LFU, and Clock policies each have advantages, no simple and accessible simulator exists for academic use. The goal is to create an interactive tool that enables students to test, compare, and visualize how replacement algorithms behave under different workloads and cache configurations.

III. Objectives

The objectives of the proposed system are as follows:

• Implement FIFO, LRU, LFU, and Clock replacement policies.
• Simulate various cache organizations (direct-mapped, set-associative, fully associative).
• Provide an educational and interactive interface for learners.
• Compare algorithmic performance based on hit ratio and execution time.

IV. Proposed System

The system architecture consists of five components: Input Module, Cache Simulator Core, Replacement Engine, Statistics Collector, and Output Visualization. The simulator accepts cache parameters and workloads, processes requests using the selected replacement policy, collects performance data, and displays graphical comparisons. The architecture promotes modularity, ease of understanding, and adaptability for academic purposes.

V. Tools and Technologies

The simulator is implemented in Python and utilizes several libraries:

• psutil – for live process and memory tracking.
• winreg – to access Windows registry for installed applications.
• collections – provides deque and dictionary structures for cache implementation.
• time – used for timestamp-based LRU operations.
• random – supports random replacement strategies.
• matplotlib – visualizes hit rate comparisons and performance metrics.

VI. Challenges and Risk Management

During development, several challenges were encountered, including restricted process access, registry permission issues, and data size variation. These were addressed by using safe access techniques, exception handling, and dynamic workload generation. Visualization challenges were mitigated through the use of Matplotlib for enhanced graphical clarity.

VII. Conclusion

The Cache Replacement Policy Simulator successfully demonstrates the comparative behavior of multiple replacement algorithms. It serves as a practical learning tool for students in understanding cache management and system performance optimization. Future work may involve adding advanced algorithms, workload import features, and web-based visualization support.

References

1. [1] A. Silberschatz, P. B. Galvin, and G. Gagne, 'Operating System Concepts', Wiley, 10th Ed., 2020.
2. [2] D. A. Patterson and J. L. Hennessy, 'Computer Organization and Design', Elsevier, 2017.
3. [3] M. D. Hill, 'DineroIV Cache Simulator', University of Wisconsin-Madison.
4. [4] 'Valgrind Cachegrind Manual', Valgrind.org.
5. [5] 'Page Replacement Algorithms in Operating Systems', GeeksforGeeks, 2024.
6. [6] IEEE Paper on Cache Replacement Policies, IEEE Xplore, 2019.