دانشگاه صنعتی خواجه نصیرالدین طوسی

K. N. TOOSI UNIVERSITY OF TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree of
BS in Computer Engineering

Page Fault Calculation

Thesis by
Fatemeh AmirabadiZadeh

Advisor
Dr. Kabeh Yaghoobi

ABSTRACT

Page replacement algorithms are critical for efficient memory management in operating systems. This project focuses on implementing and analyzing three widely used algorithms: First-In-First-Out (FIFO), Least Recently Used (LRU), and Optimal Page Replacement (OPT). By simulating these algorithms, we aim to calculate and compare page faults under varying workloads and memory constraints. The insights gained highlight the trade-offs and efficiency of each algorithm, contributing to a deeper understanding of memory management in operating systems.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

C h a p t e r 1

INTRODUCTION

1.1 Objective

The primary objective of this project is to study the behavior and performance of three page replacement algorithms: FIFO, LRU, and OPT. Through implementation and simulation, we aim to calculate the number of page faults for each algorithm under various reference strings and memory frame sizes. By comparing the results, we seek to analyze their strengths, weaknesses, and practical applications in memory management. This project also aims to enhance understanding of the relationship between memory constraints and algorithm efficiency.

1.2 Methodology

The problem addressed in this project involves simulating a system with limited memory frames and determining how different algorithms handle page faults. The input consists of a sequence of page references, known as the reference string, and the number of available memory frames. The implementation was carried out using Python on Linux operating system. Arrays and additional data structures, such as hash maps and tables, were used to support the requirements of each algorithm.

Chapter 2

FIFO

The First-In-First-Out (FIFO) algorithm operates by replacing the oldest page in memory when a page fault occurs. This is achieved by maintaining a queue where pages are inserted as they are referenced. When a replacement is necessary, the page at the front of the queue is removed, making room for the new page. FIFO is straightforward to implement but can suffer from the anomaly known as Belady's Anomaly, where increasing the number of frames may increase page faults.

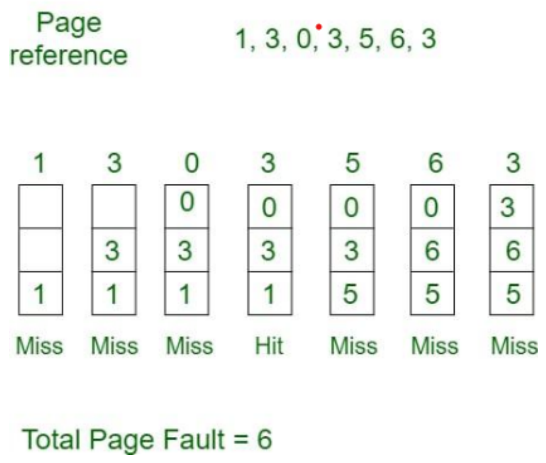

Figure 2.1: FIFO Page Replacement Algorithm

Chapter 3

LRU

The Least Recently Used (LRU) algorithm replaces the page that has not been used for the longest period. This is accomplished by tracking the usage history of each page. In our implementation, a data structure such as a stack or a hash map was used to maintain the order of page references. LRU tends to perform better than FIFO in many cases because it takes into account the actual usage pattern of pages, but it is computationally more expensive due to the need to update the usage history frequently.

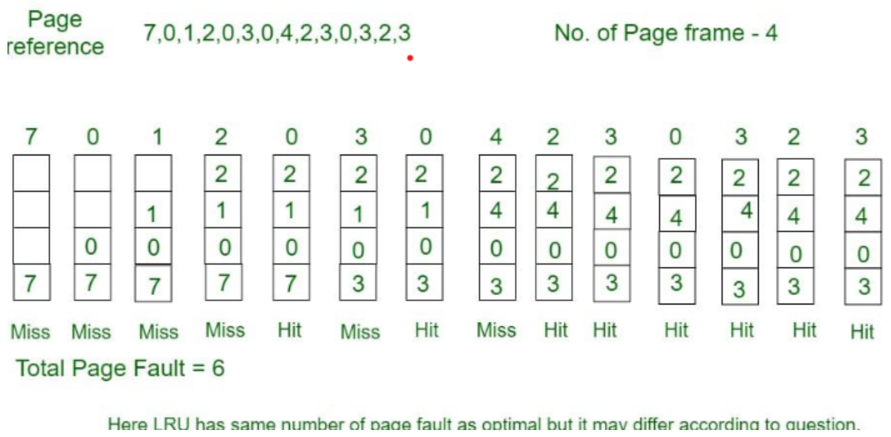
Page reference
7,0,1,2,0,3,0,4,2,3,0,3,2,3
No. of Page frame - 4
7 0 1 2 0 3 0 4 2 3 0 3 2 3
Miss Miss Miss Miss Hit Miss Hit Miss Hit Hit Hit Hit Hit Hit
Total Page Fault = 6
Here LRU has same number of page fault as optimal but it may differ according to question.

Figure 3.1: LRU Page Replacement Algorithm

C h a p t e r 4

OPT

The Optimal Page Replacement (OPT) algorithm replaces the page that will not be used for the longest time in the future. This algorithm requires knowledge of future page references, which makes it impractical for real-time systems but ideal for theoretical comparisons. In the implementation, the reference string was scanned ahead to determine the page that would be used furthest in the future, and that page was replaced during a page fault. OPT provides the least number of page faults and serves as a benchmark to evaluate other algorithms.

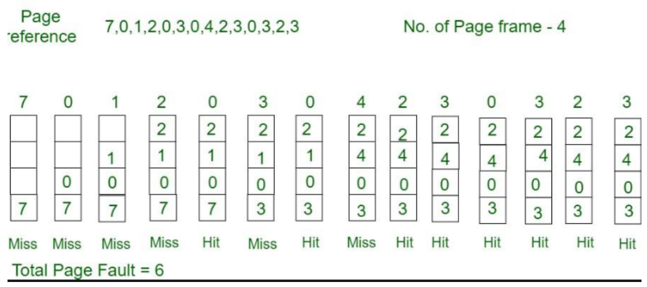

Figure 4.1: OPT Page Replacement Algorithm

Chapter 5

CONCLUSION

This project provides an in-depth analysis of FIFO, LRU, and OPT page replacement algorithms by implementing and simulating their behavior in a controlled environment. The results show that each algorithm has unique strengths and limitations. FIFO is simple but may result in suboptimal performance due to its lack of consideration for usage patterns. LRU performs well in most scenarios by leveraging historical usage data but requires additional computational overhead. OPT, while theoretically optimal, is impractical for real-world applications due to its dependence on future knowledge. These findings underscore the importance of selecting appropriate page replacement strategies based on specific use cases and constraints.

Chapter 6

FUTURE WORK

There is significant potential for extending this project. Future work could include the implementation of additional page replacement algorithms, such as Clock or Second-Chance, to further explore their performance characteristics. Real-world workloads could be used to evaluate algorithm efficiency under practical conditions. Additionally, techniques to optimize cache performance and reduce page fault rates could be investigated to enhance system performance in modern computing environments.

Appendix A

REFERENCES

1. GeeksforGeeks. (n.d.). *page replacement algorithms in operating systems*. Retrieved from [https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/]

2. Scaler. (n.d.). *LRU Page Replacement Algortihm*. Retrieved from [https://www.scaler.com/topics/lru-page-replacement-algorithm/]

INDEX