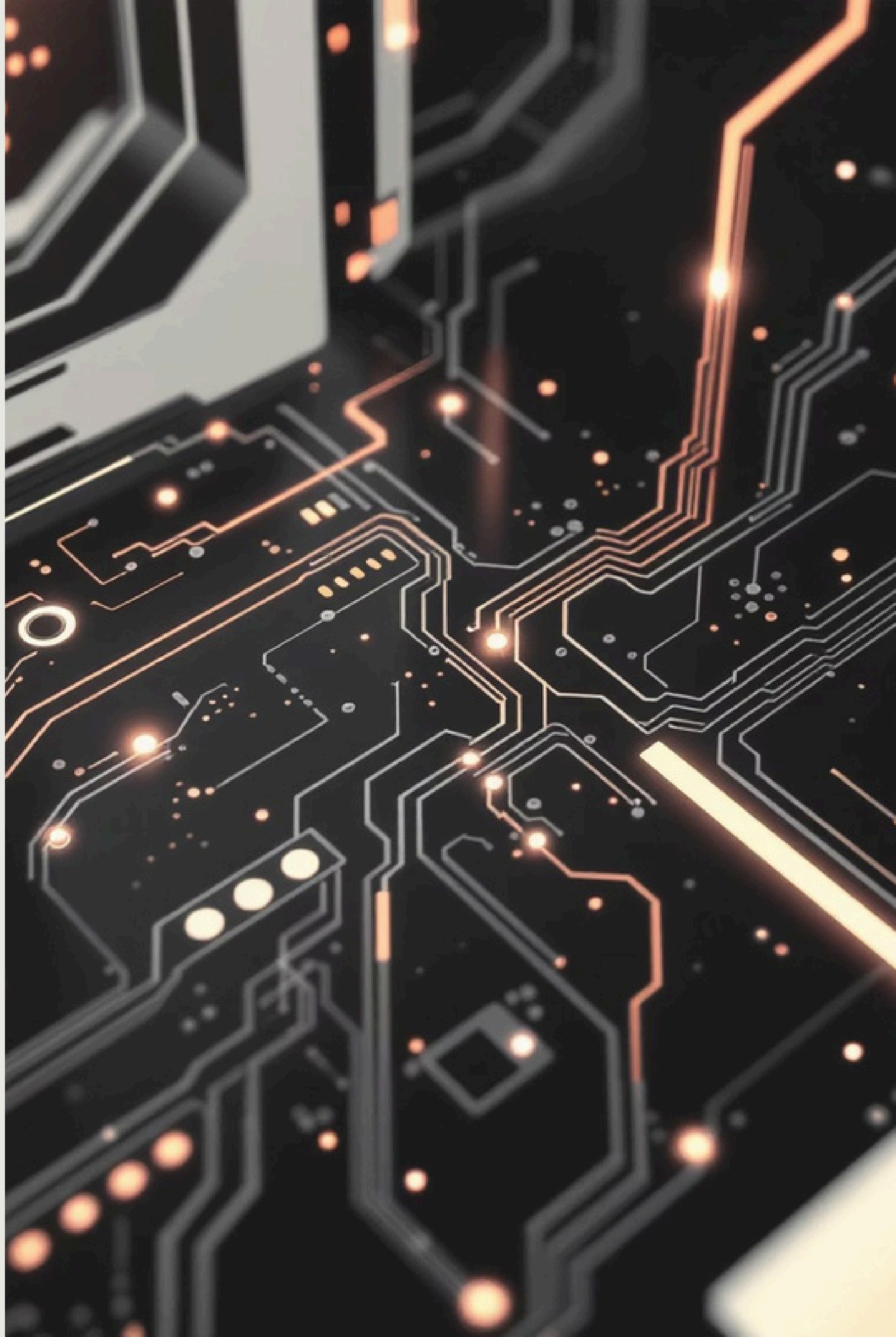
VIRTUAL MEMORY MANAGEMENT SIMULATOR

Yash Jadhav 3RollNumber:2303139

Group Name: PageFault Squad

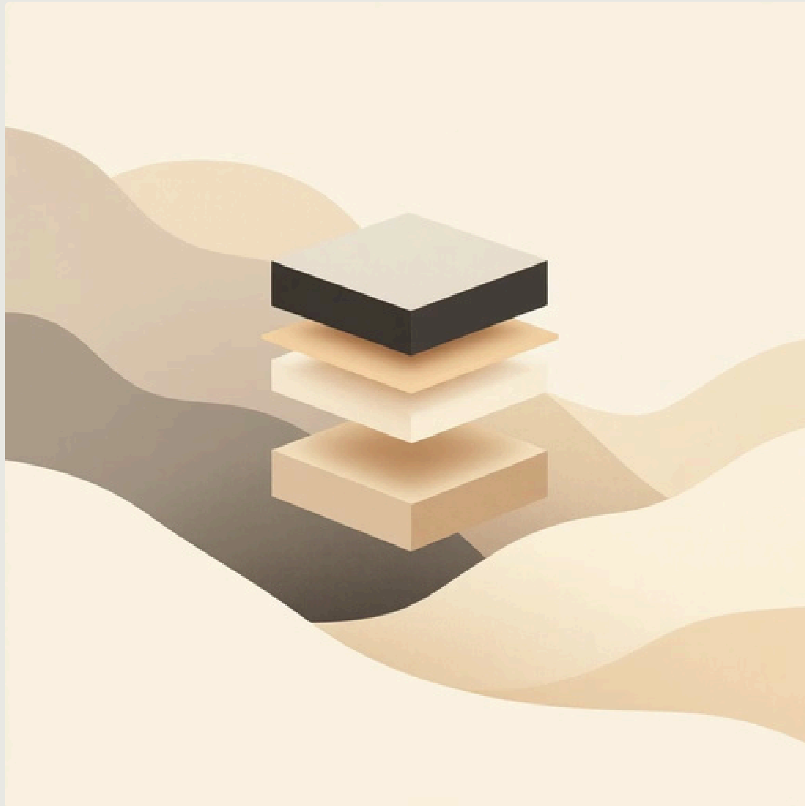


INTRODUCTION TO VIRTUAL MEMORY SIMULATION

Virtualmemoryis a fundamental operating system technique thatallows programs to use more memory than is physically available. It abstracts and manages memory resources, providing a contiguous address space to each process.

Simulating virtual memory is crucial for understanding its complex mechanisms, such as paging, segmentation, and page replacement algorithms, without directly manipulating a live system. It offers a safe environment for experimentation and analysis.

Our project aims to provide a robust, interactive simulation environment for virtual memory management. The goal is to visualize the performance of various paging and replacement algorithms, offering insights into their efficiency and behavior.



PROJECT OVERVIEW

WINDOWS-BASED ENV IRONMENT

Developed torunseamlessly onthe Windows operating system, ensuring broad accessibility for users.

PYTHONTKINTER GUI

Anintuitive graphical userinterface built with Python's Tkinter for user input and clear display of results.

C++ BACKEND

High-performance C++ implementation handles the core paging simulation logic and complex algorithm execution.

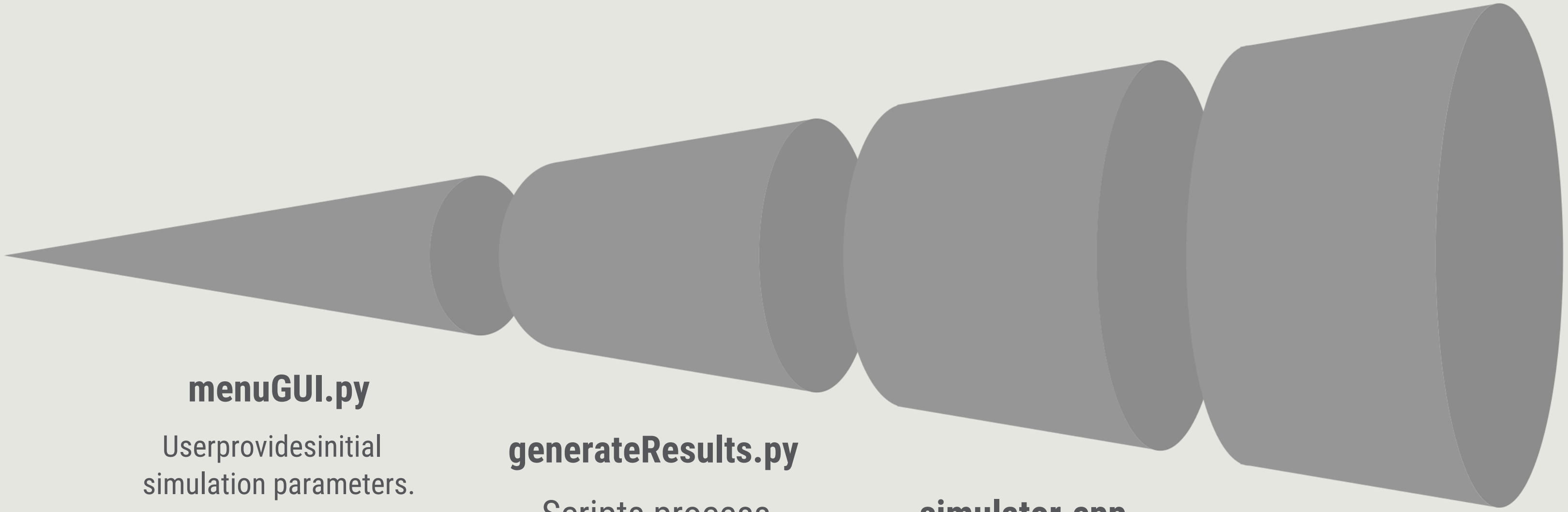
AUTOMATED PLOT GENERATION

Automaticallygeneratesinsightful graphs to visually represent simulation outcomes and performance metrics.

ALGORITHM VERSATILITY

Supportsawide arrayof paging and page replacement algorithms, allowing for comprehensive comparative analysis.

SYSTEM ARCHITECTURE DIAGRAM



menuGUI.py

Userprovidesinitial simulation parameters.

generateResults.py

Scripts process inputs and configure simulation.

simulator.cpp

Core engine runs the complex simulation.

resultsGUI.py

Final data is visualized in the results interface.

This architecture provides a clear separation of concerns, from user interaction to core simulation logic and result visualization.

KEY FEATURES

GUIFEATURES

- User-friendlyinput forms
- Real-time progress updates
- Interactive result displays

C++ SIMULATION

- Efficient pagingsimulation
- Precise algorithm execution
- Detailed performance metrics

GRAPH GENERATION

- Automated plotcreation
- Visual data analysis
- Customizable graph parameters

ALGORITHMS IMPLEMENTED

FETCH POLICIES

- ### DEMAND PAGING

  Pagesare onlyloadedinto memory when they are actually referenced, reducing initial memory overhead.

- ### PRE-PAGING

  Anticipates futurepage accesses and loads them into memory before they are explicitly demanded, aiming to reduce page faults.

REPLACEMENT POLICIES

- ### FIFO (FIRST-IN, FIRST-OUT)

  The oldestpageinmemoryisreplacedfirst,regardless of how often it's used.

- ### LRU (LEAST RECENTLY USED)

  Thepage that hasnotbeenusedfor thelongestperiod is replaced, based on the assumption that past usage predicts future usage.

- ### SECOND CHANCE (CLOCK)

  Amore efficientvariant ofFIFOthatgivespages a "second chance" before being replaced if they have been accessed recently.

- ### OPTIMAL REPLACEMENT

  Replaces thepage thatwill not beusedfor the longest period of time in the future. Serves as a benchmark for comparison.

DETAILED ALGORITHM DESCRIPTIONS

1

DEMAND PAGING MECHANICS

Apage faulttriggers whenareferencedpageisnotinmemory. The OS then loads the required page from secondary storage, minimizing memory footprint and speeding up process startup.

2

PRE-PAGING ADVANTAGE

Byloadingpagesproactively, pre-paging aims to reduce the number of page faults during subsequent execution, potentially improving overall system performance in scenarios with predictable access patterns.

3

FIFO VS. LRU DECISIONS

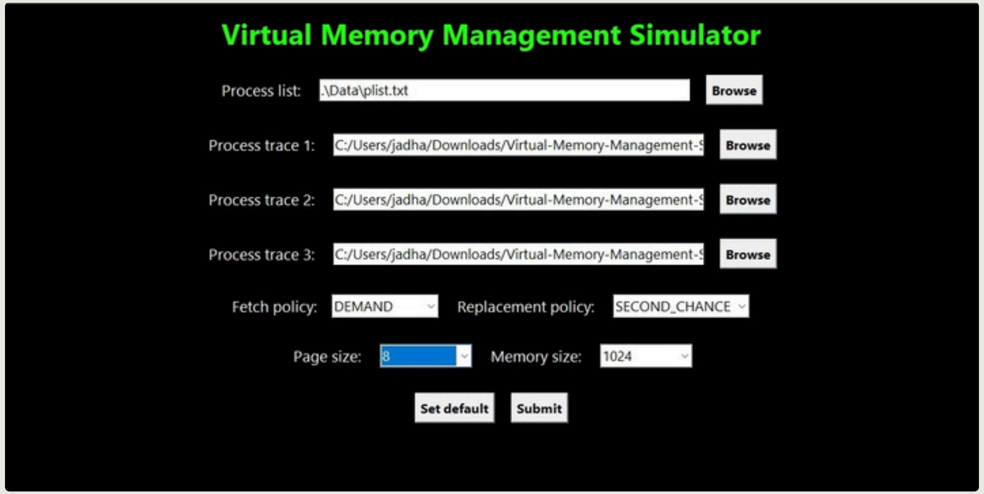
FIFO is simpletoimplementbut can discardfrequently used pages. LRU, while more complex due to tracking usage, generally achieves better performance by retaining more relevant pages.

4

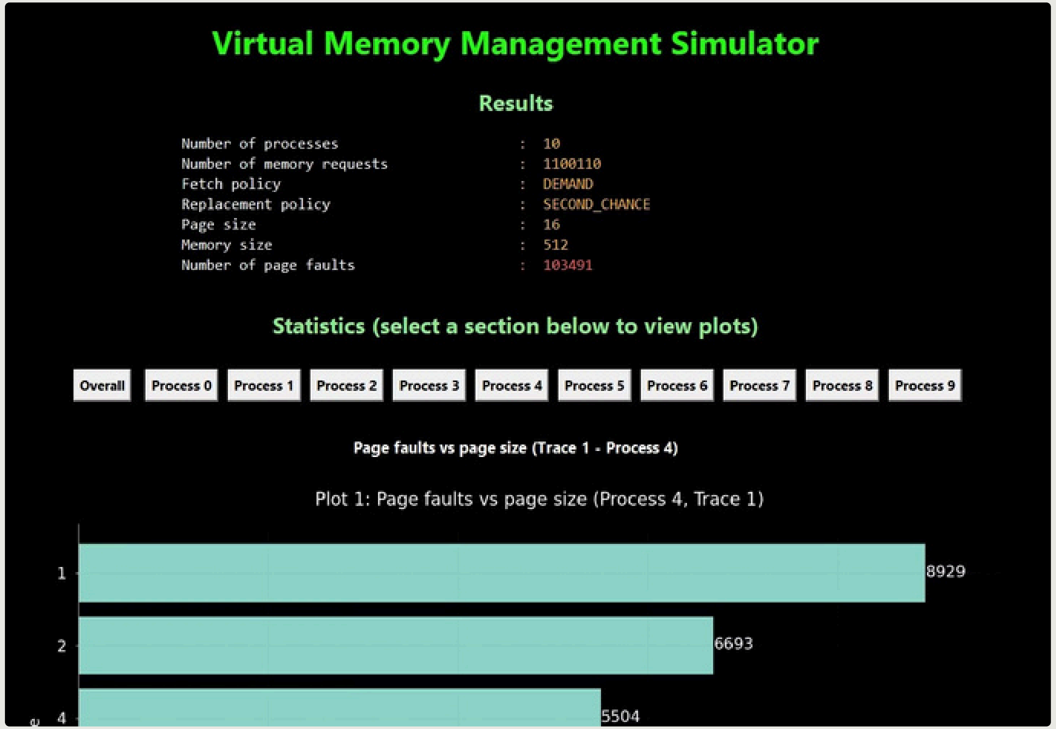
OPTIMAL: THE THEORETICAL BEST

TheOptimal algorithm is impossibleto implementinpractice as it requires knowing the future. However, it provides an invaluable theoretical lower bound for the number of page faults, against which other algorithms are measured.

GUI SHOWCASE

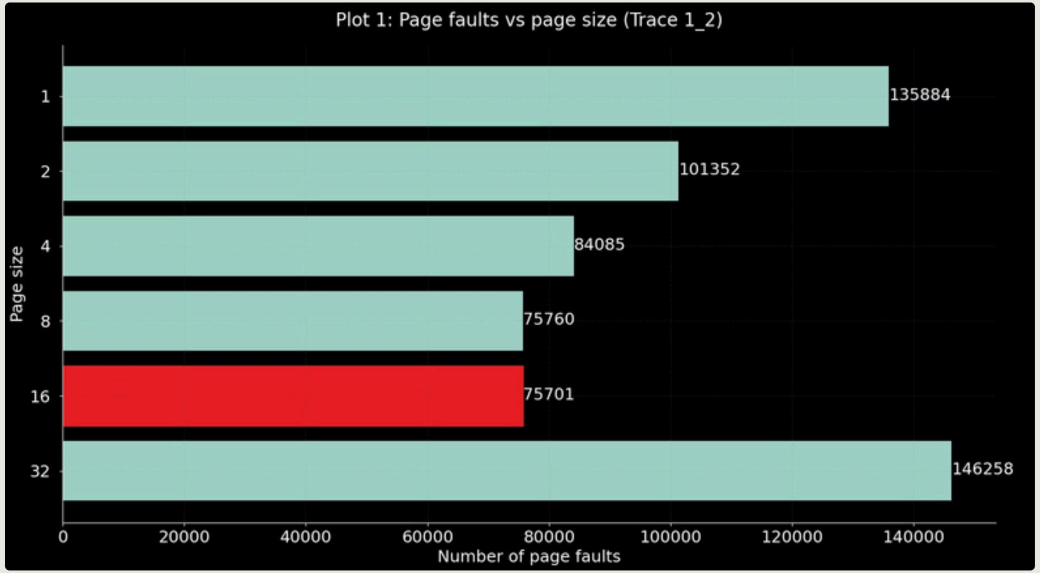


INPUT GUI



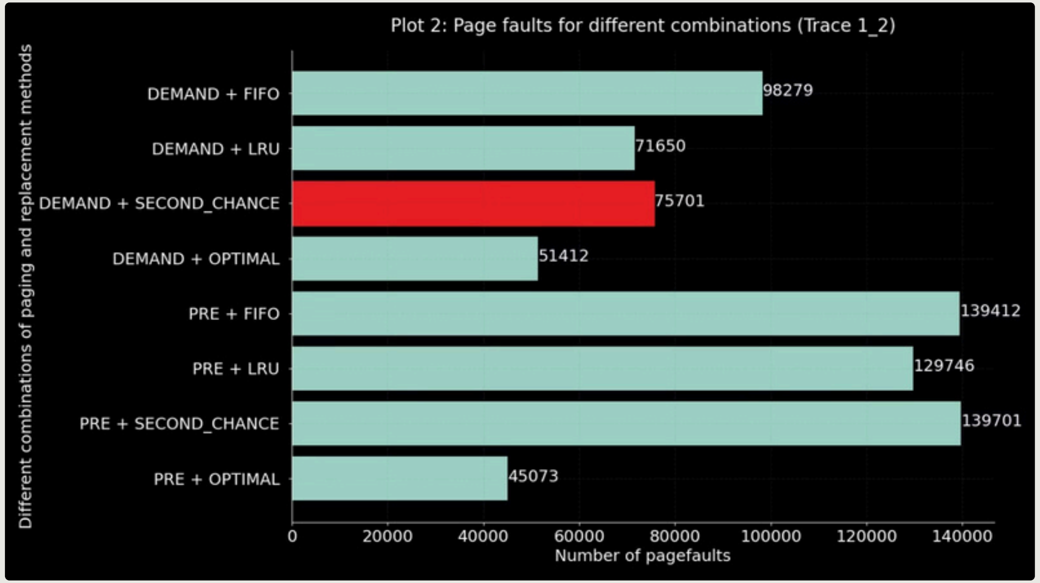
RESULTS GUI

SIMULATION OUTPUT VISUALIZATIONS



PLOT 1: PAGE SIZE VS PAGE FAULTS

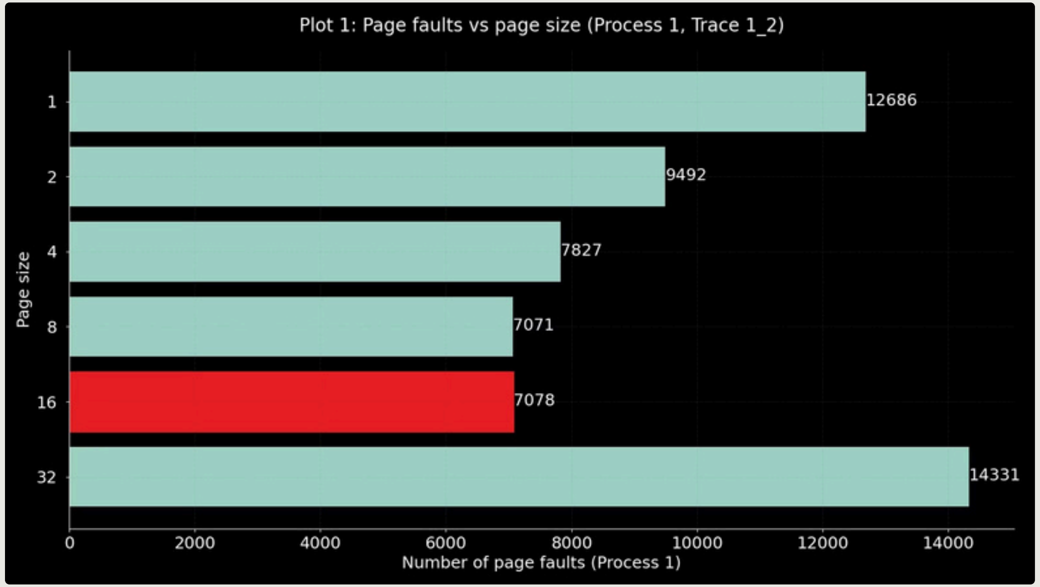
Thisplot illustrateshowvaryingpagesizesimpactthenumber of page faults, a critical metric for memory management efficiency.



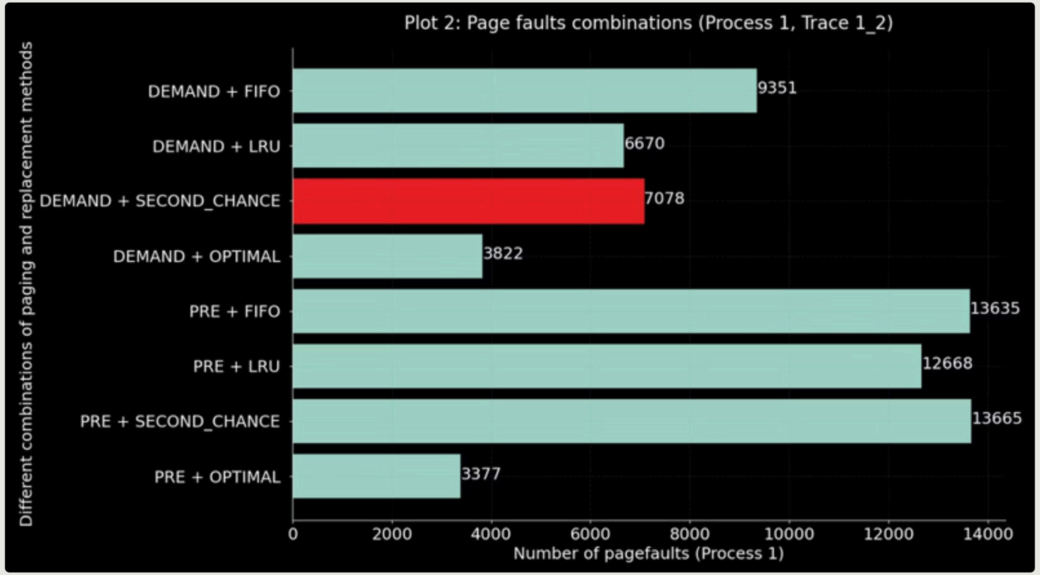
PLOT 2: POLICY COMBINATIONS VS PAGE FAULTS

Here,we comparetheeffectivenessof differentpagingandreplacement algorithm combinations in minimizing page faults.

SIMULATION OUTPUT VISUALIZATIONS - PER PROCESS



PLOT 1: PAGE SIZE VS PAGE FAULTS



PLOT 2: POLICY COMBINATIONS VS PAGE FAULTS

ENHANCEMENTS IMPLEMENTED

INDIVIDUAL PER-PROCESS PLOTS

Visualizeperformanceandmetricsforeachprocessindependently to gain granular insights.

REAL-TIME DATA STREAMING

Process and displayPtracedatainreal-timebatchesfor immediate analysis and responsiveness.

CONCLUSION & FUTURE ENHANCEMENTS

SIMULATOR DEMONSTRATES

Oursimulator effectively models virtualmemory behavior and highlights the trade-offs between various paging strategies.

ADVANTAGES OF VISUALIZATION

Visualizing pagefaults and memory statessimplifiesunderstanding complex OS concepts and algorithm performance.

EDUCATIONAL IMPACT

Thistoolservesasan invaluable educational aid for students and researchers exploring operating systems.

FUTURE ENHANCEMENTS

Consideradding supportfor segmentation, multi-level paging, and more advanced replacement algorithms for deeper analysis.