OPERATING SYSTEM PROJECT
PAGE REPLACEMENT USING PSUTIL

22BRS1006 NAGA RITHESH
22BRS1147 SP ROOPESH

Abstract:

The management of virtual memory is a critical aspect of modern computer operating systems, allowing programs to exceed the physical memory capacity by utilizing disk storage as an extension. Page replacement algorithms play a pivotal role in optimizing memory usage and minimizing page faults, thereby enhancing overall system performance. This project explores the implementation and simulation of three prominent page replacement algorithms—LRU (Least Recently Used), FIFO (First-In-First-Out), and Optimal—using the psutil and pandas libraries in Python.

The code utilizes psutil to monitor and gather information about running processes, focusing on details such as process ID, CPU usage, memory usage, and more. Three page replacement algorithms are implemented to simulate the eviction and replacement of pages in memory, providing insights into their effectiveness in managing virtual memory. The algorithms are assessed based on their impact on overall system performance and memory utilization.

Key concepts covered in the project include process monitoring, data analysis with pandas, datetime manipulation for timestamp handling, and the use of classes and objects to encapsulate functionality. The PageReplacementSimulator class orchestrates the simulation, tracking page replacement history and providing methods to display process information and recent replacement actions.

The project concludes by demonstrating the effectiveness of the implemented algorithms in optimizing memory usage and minimizing page faults. The combination of psutil, pandas, and simulation techniques offers a comprehensive understanding of page replacement strategies and their implications on system performance.

PAGE REPLACEMENT ALGORITHMS

Page replacement algorithms play a crucial role in managing memory allocation when a new page is required. A variety of algorithms have been developed to handle page replacement efficiently, with the ultimate goal of minimizing page faults. By minimizing page faults, these algorithms enhance the overall performance of the process.
In computer operating systems, page replacement is a crucial aspect of managing a system's virtual memory. Virtual memory is a memory management technique that creates an illusion of a large main memory by utilizing disk storage as an extension of physical memory.
When a computer system employs virtual memory, it allows programs to be larger than the available physical memory (RAM). To handle this scenario, the operating system transfers data

between the physical memory and the disk, enabling the illusion of a larger memory space. The basic unit of data transfer between the disk and the main memory is known as a "page."
Page replacement refers to the process of selecting a page in memory that is not currently being used and replacing it with a page from the disk. This situation arises when a program attempts to access a page that is not present in the main memory, resulting in a "page fault." In such cases, the operating system must determine which page in the main memory should be replaced with the required page from the disk.
Various page replacement algorithms have been developed, each with its own set of advantages and disadvantages. These algorithms are designed to optimize the performance of memory management. Some common page replacement algorithms include:

1. FIFO (First-In-First-Out): This algorithm replaces the oldest page, which was brought into the main memory first.
2. LRU (Least Recently Used): This algorithm replaces the page that has not been accessed for the longest duration.
3. LFU (Least Frequently Used): This algorithm replaces the page that has been accessed the fewest number of times.
4. Optimal: This theoretical algorithm replaces the page that will not be used for the longest duration in the future. However, it is not practical to implement in real-time scenarios due to the need for future knowledge.

The choice of a page replacement algorithm depends on various factors, including the system's workload, the characteristics of the applications running on the system, and the available hardware resources. Each algorithm has its own trade-offs in terms of complexity and performance.
By employing efficient page replacement algorithms, computer systems can effectively manage virtual memory, optimize memory usage, and enhance overall system performance.

USE OF PSUTIL
psutil is a cross-platform library for retrieving information on running processes and system utilization, including memory usage. While psutil itself doesn't provide direct support for page replacement algorithms, you can use it to gather information about the memory usage of processes and the system as a whole. With this information, you can implement your own logic for page replacement.
We are extracting the pages replaced during the process.

CONCEPTS LEARNT
Process Monitoring: The code uses the psutil library to monitor and gather information about running processes on the system. It retrieves details such as process ID, name, CPU usage, memory usage, I/O counters, status, creation time, number of threads, and CPU affinity.
Page Replacement Algorithms: The code implements three page replacement algorithms: LRU (Least Recently Used), FIFO (First-In-First-Out), and Optimal. These algorithms are used in operating systems to manage memory and decide which pages to evict from memory when a new page needs to be loaded.

Simulating Page Replacement: The PageReplacementSimulator class simulates the page replacement process by adding processes, updating page frames based on the chosen algorithm, and performing page replacements. It keeps track of the page replacement history and provides methods to display the actions taken.
Data Analysis with Pandas: The code uses the pandas library to organize and analyze process information. It creates a DataFrame to store process data and provides methods to print process information and display a process viewer.
Datetime Manipulation: The code uses the datetime module to handle timestamps and calculate time differences. It converts timestamps to datetime objects and performs operations such as subtracting time intervals.
Class and Object: The code defines two classes, Process and PageReplacementSimulator, to encapsulate related data and functionality. Objects of these classes are created and used to represent processes and simulate page replacement.
OrderedDict: The code uses OrderedDict from the collections module to maintain the order of page frames and page replacement history. It ensures that the order of elements is preserved when iterating or performing operations on the dictionary

CONCLUSION

The code defines two classes: Process and PageReplacementSimulator. The Process class represents a process with various attributes such as PID, name, CPU usage, memory usage, etc. The PageReplacementSimulator class simulates page replacement algorithms like LRU (Least Recently Used), FIFO (First-In-First-Out), and Optimal.
The code initializes an instance of the PageReplacementSimulator class and adds processes using the psutil library. It then simulates page replacement using the LRU, FIFO, and Optimal algorithms. The process information is printed using the print_process_info method, and the process viewer is displayed using the display_process_viewer method. Finally, the recent page replacement actions during the execution of the current process are printed using the get_recent_replacement_actions method.
In conclusion, this code simulates page replacement algorithms using the psutil and pandas libraries in Python. It provides functionality to add processes, simulate page replacement using different algorithms, display process information, and view process details.