Makara Ramoabi

20240045

Operating system

1. Scenario Analysis

In a cloud computing platform such as Amazon Web Services (AWS), multiple virtual machines run on shared physical servers. Each virtual machine believes it has its own large, continuous block of memory, even though the actual physical RAM is limited and shared among many users. This is made possible through virtual memory. When an application inside a virtual machine requests memory, the operating system maps virtual addresses to physical memory using page tables. If the required page is not in RAM, a page fault occurs, and the system loads the page from disk storage into memory. Less frequently used pages may be moved back to disk to free space. This allows cloud providers to run many applications efficiently while maintaining isolation and security between users. Virtual memory improves resource utilization, scalability, and system stability in high-demand cloud environments.

2. Concept Research

The Least Recently Used (LRU) algorithm is a page replacement strategy used in virtual memory systems. When physical memory is full and a new page must be loaded, the operating system must decide which existing page to remove. LRU selects the page that has not been used for the longest period of time. The assumption is that pages used recently are more likely to be used again soon. By replacing the least recently accessed page, LRU reduces the number of page faults and improves overall system performance. It is widely used because it balances efficiency and simplicity in memory management.

3. Tool Practice

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;


int main() {
```

```
    vector<int> pages = {1, 2, 3, 2, 4, 1, 5};
    vector<int> memory;
    int capacity = 3;
    int pageFaults = 0;

    for (int page : pages) {
        auto it = find(memory.begin(), memory.end(), page);

        if (it == memory.end()) {
            pageFaults++;
            if (memory.size() == capacity) {
                memory.erase(memory.begin()); // Remove oldest (simple LRU simulation)
            }
            memory.push_back(page);
        } else {
            memory.erase(it);
            memory.push_back(page); // Move page to most recent position
        }

        cout << "Current Memory: ";
        for (int m : memory) cout << m << " ";
        cout << endl;
    }

    cout << "Total Page Faults: " << pageFaults << endl;
    return 0;
}
```

Reflection

Writing this C++ simulation in Visual Studio Code helped me understand how virtual memory and page replacement algorithms work. The program demonstrates a simplified LRU strategy by tracking recently used pages in a vector. When memory reaches capacity, the least recently used page is removed. Running the program allowed me to see how page faults occur and how memory content changes over time. This hands-on activity made the theoretical concept of LRU clearer and easier to visualize. It also improved my understanding of memory management, page replacement policies, and how operating systems optimize limited physical memory resources.

4. Diagram Design

The memory diagram should show three main components: the CPU, physical memory (RAM), and secondary storage (disk). Virtual addresses generated by the CPU are translated using a page table into physical addresses in RAM. If the requested page is present in RAM, execution continues normally. If not, a page fault occurs, and the operating system loads the required page from disk into memory. The diagram should also show pages being swapped out when memory is full. Arrows illustrate address translation and data movement. This visual representation clearly explains demand paging and how virtual memory manages limited physical resources efficiently.