Project Report

Introduction

In this project, I developed a virtual memory system simulator to study the performance of different page replacement algorithms, namely First-In-First-Out (FIFO), Least Recently Used (LRU), and Segmented FIFO. I used two trace files, bzip.txt and sixpack.txt, each containing 1 million memory traces, and tested the algorithms with different numbers of page frames. The goal was to analyze the performance of each algorithm and understand the impact of available memory on the system's performance.

Methods

I performed simulations with varying frame sizes: 1, 2, 3, 5, 8, 13, 21, 34, 55, and 89. These frame sizes were chosen as they represent a range of memory allocations, from insufficient memory to more than the required memory for the processes. I also analyzed the disk reads and disk writes for each algorithm, using both trace files, to determine the performance of the algorithms and draw comparisons between them.

Results

The results obtained from the simulations show varying performance of the page replacement algorithms across different memory sizes. To visualize the performance of each algorithm, I created charts that plot the number of disk reads and writes as a function of the number of frames. The following observations can be made:

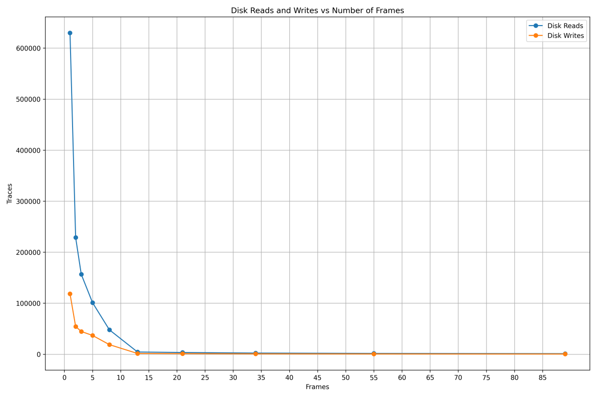

Fig 1.1 FIFO bzip.trace

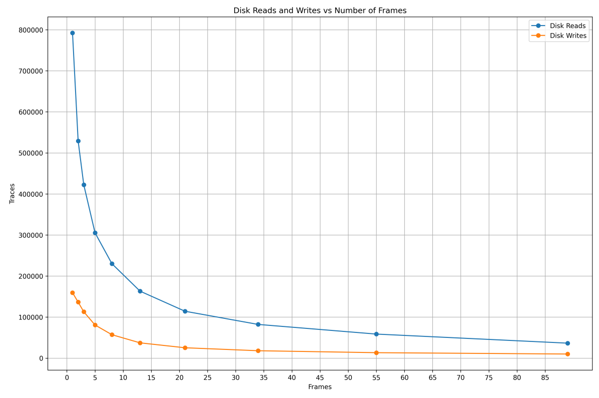

Fig 1.2 FIFO sixpack.trace

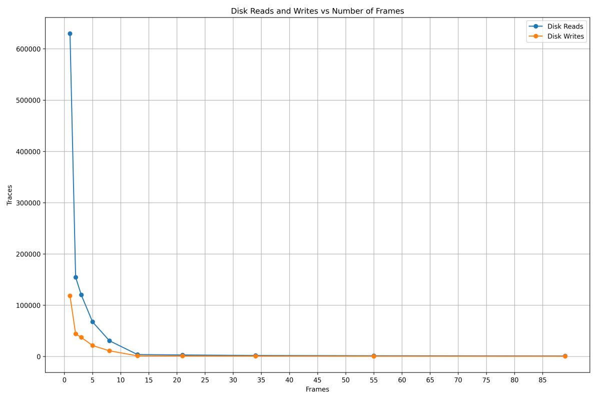

Fig 2.1 LRU bzip.trace

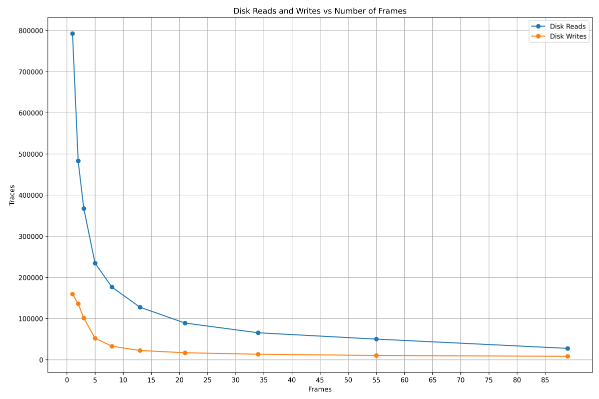

Fig 2.2 LRU sixpack.trace

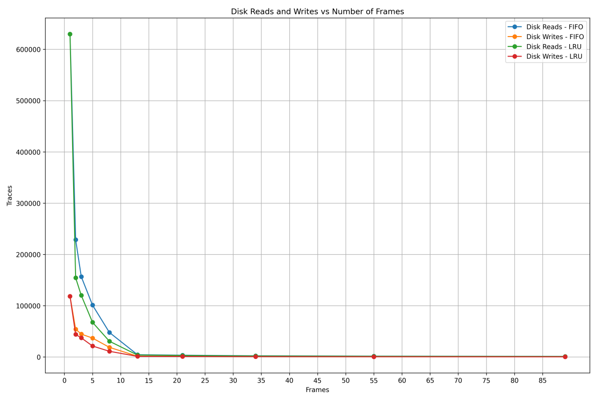

Fig 3.1 FIFO vs LRU bzip.trace

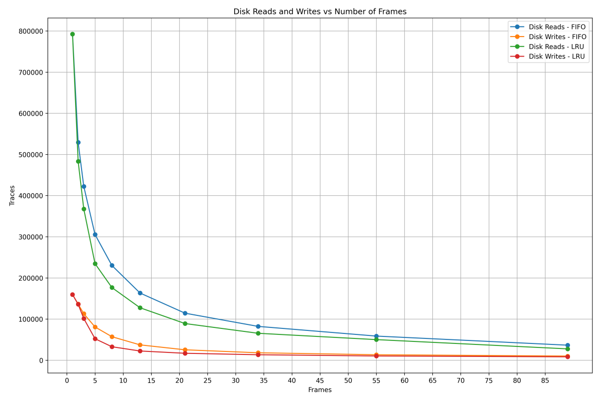

Fig 3.2 FIFO vs LRU sixpack.trace

- For both trace files, as the number of frames increases, the number of disk reads and writes generally decreases for all algorithms. This trend can be clearly observed in the charts, where the lines representing disk reads and writes for each algorithm consistently decrease as the number of frames increases. This is expected, as having more memory available reduces the need for page replacements.
- LRU consistently performs better than FIFO in terms of disk reads and writes for both trace files, as evident from the charts. The lines representing LRU are consistently below the lines for FIFO, indicating that it is a more efficient page replacement algorithm.
- The performance difference between the two trace files suggests that the processes have different memory requirements and working set sizes. This can be observed in the fact that the number of disk reads and writes differs between the trace files for the same frame sizes and algorithms. The charts also show different slopes and patterns for the two trace files, further supporting this observation.

Conclusions:

From the experimental results, we can conclude that:

- The LRU algorithm generally performs better than the FIFO algorithm for the given memory traces.
- The size of available memory has a significant impact on the performance of the memory system. As the memory size increases, the number of disk reads and writes decreases, indicating better performance.
- The working set size and memory requirements of the processes in the trace files are different, as evidenced by the variations in performance across the trace files.

These conclusions provide insights into the behavior of different page replacement algorithms and the importance of considering memory allocation and working set sizes when optimizing the performance of memory systems.