Project Part 3: Page Table Algorithm Comparison Report

Neiloy Hossain

1. Introduction

The goal of this experiment was to compare different types of page replacement methods (First in First Out, Least Recently Used, and Least Frequently Used) and see their performance, and if there is a pattern in hit rate, miss rate, and page faults. The key independent variables that changed are the page replacement algorithms and memory access patterns. The dependent variables that we used to measure are the hit rate, miss rate, and how many page faults that have occurred. The reference patterns that were used:

- Random - Pages that were accessed randomly
- Locality - Accesses in clusters in small sets, repeatedly
- Sequential - Accesses within numerical order

2. Key Observations

The algorithm that performed the best based on the hit rate, miss rate, and page faults was the LFU algorithm. It was very dependent on the reference pattern, as the LRU algorithm performed best under locality patterns, while the LFU algorithm was best under random and sequential patterns.

On the random reference pattern, all algorithms were similar, but the LFU algorithm performed better than the FIFO and LRU algorithms, where there were issues of having a lower hit rate and higher miss rate. For the locality reference pattern, the LRU Algorithm has performed the best, having a higher hit rate and the lowest miss rate. The algorithm that performed the worst was LFU, with its lower hit rate and higher page faults. When the sequential reference pattern was used, the LFU algorithm performed the best, and the FIFO and LRU algorithms performed the worst. It seems like the FIFO algorithm performed the worst out of all patterns, where the LRU algorithm was best with the locality reference pattern, and the LFU algorithm was best with the sequential reference pattern.

3. Reflection

The LRU algorithm performed well with locality because it kept a small set of pages and was used repeatedly. The FIFO algorithm did not perform so well because of how it ran via the load order. For this algorithm, it does not matter about the usage at all, causing it to affect its performance. The LFU's frequency counting affects its performance in these scenarios by when the frequently accessed pages are in memory. It will also be affected negatively by accessing new pages, and if pages are no longer needed.

These results relate to real-world program behavior and memory access by simulating the locality reference pattern as users and applications tend to use a small set of memory, like in loops. From looking at the charts, we can see that the LRU algorithm works best to match this behavior. The best-performing algorithm is never the most practical choice, like the LRU algorithm often performs the best, but it has to track when all the pages were last used. The FIFO algorithm uses less memory and CPU resources, but the caveat is that it is pricey from a performance perspective. The LFU algorithm uses more memory and more overhead to maintain its frequency.