Kavya Sharma

CSEN 177

Lab 8

**Lab Report**

**FIFO (First In, First Out):**

This algorithm implements a simple page replacement policy where the oldest page in the cache is replaced when a new page needs to be loaded. Pages are maintained in a circular queue, with a pointer (placeInArray) used to track the next replacement. Results in a page fault whenever a requested page is not in the cache and an old page needs to be replaced.

I noticed that FIFO is a simpler algorithm but often replaces frequently used pages, leading to higher fault rates.

**LRU (Least Recently Used):**

This algorithm tracks the usage of pages using an "age" or "index" field to represent when a page was last accessed.On a cache hit, the age of the accessed page is reset, while all other pages' ages are incremented.The least recently used page (i.e., the one with the highest age) is replaced when needed.

I noticed that LRU achieves better hit rates as it specifically targets the least recently used page for replacement, aligning more closely with real-world access patterns.

**Second Chance:**

This algorithm builds upon FIFO but gives pages a "second chance" if accessed recently (indicated by a chance bit).When a page fault occurs, the algorithm cycles through pages circularly.If a page has a chance bit set, it resets the bit and skips replacement.If a page's chance bit is unset, it is replaced.

I noticed that Second Chance strikes a balance between FIFO's simplicity and LRU's efficiency but does not outperform LRU due to its reliance on approximations (chance bits).

Table representing numReqs, numFaults, Miss Rate and Hit Rate:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | Cache Size | numReqs | numFaults | Miss Rate | Hit Rate |
| FIFO | 10 | 10,000 | 9916 | 0.9916 | 0.0084 |
|  | 50 | 10,000 | 9515 | 0.9515 | 0.0485 |
|  | 100 | 10,000 | 9018 | 0.9018 | 0.0982 |
|  | 250 | 10,000 | 7534 | 0.7534 | 0.2466 |
|  | 500 | 10,000 | 5130 | 0.5130 | 0.487 |
| LRU | 10 | 10,000 | 9915 | 0.9915 | 0.0085 |
|  | 50 | 10,000 | 9510 | 0.9510 | 0.049 |
|  | 100 | 10,000 | 9029 | 0.9029 | 0.0971 |
|  | 250 | 10,000 | 7532 | 0.7532 | 0.2468 |
|  | 500 | 10,000 | 5206 | 0.5206 | 0.4794 |
| Second Chance | 10 | 10,000 | 9915 | 0.9915 | 0.0085 |
|  | 50 | 10,000 | 9510 | 0.9510 | 0.0490 |
|  | 100 | 10,000 | 9022 | 0.9022 | 0.0978 |
|  | 250 | 10,000 | 7526 | 0.7526 | 0.2474 |
|  | 500 | 10,000 | 5178 | 0.5178 | 0.4822 |

Graph of Hit Rate v/s Cache Size:

A graph with a line graph

Description automatically generated

We can see that all three algorithms display an increasing hit rate as the cache size grows, which is expected since larger caches can accommodate more pages, reducing the likelihood of faults. The hit rate approaches a value closer to 50% for the largest cache size (500).

The FIFO (First-In-First-Out) algorithm generally underperforms compared to LRU and Second Chance at smaller cache sizes, reflecting its simpler replacement strategy. The LRU (Least Recently Used) algorithm shows slightly better performance than FIFO for intermediate cache sizes, benefiting from its more adaptive replacement strategy. The Second Chance algorithm performs very similarly to LRU, especially for larger cache sizes, since it combines FIFO characteristics with the ability to retain frequently accessed pages.

For small or medium cache sizes, LRU and Second Chance are more effective at minimizing faults due to their intelligent replacement policies. For large cache sizes, the difference between algorithms becomes negligible, suggesting that simpler policies like FIFO might be sufficient.