CS314: Lab Assignment 8

B Siddharth Prabhu
`200010003@iitdh.ac.in`

19 March 2023

First, we will briefly go through expected characteristics of the algorithms. Then, we shall look at observed behaviour.

1 FIFO Page Replacement Policy

In this policy, we keep track of all pages in the memory in a queue. Hence, the oldest page is in the front of the queue. When a page needs to be replaced, the page in the front of the queue is selected for eviction.

1.1 Running the code

Run the bash script provided in the following format:
`bash run_fifo.sh <arg1> <arg2> <arg3> <arg4>` , where the arguments are exactly as described in the problem statement. (arg1 is number of addressable pages in the system, arg2 is number of pages/frames in main memory, arg3 is number of pages in swap space of disk, and arg4 is input file name)

1.2 Advantages

- Ease of Implementation: This algorithm is quite simple and straightforward, making it a popular choice.
- Fairness: Since the oldest page is evicted, each page is effectively at the same priority, and will surely be evicted after some time. This ensures fairness.
- Low Data Structure Overhead: Queue is quite easy to implement, so it has a low computation overhead. Accesses involve dequeueing from the front of the queue for eviction, and enqueueing in the back of the queue for new entries.

1.3 Limitations

- Poor Performance: If pages are addressed in a manner such that the one just evicted is requested again, in a looping fashion, then it performs badly.
- Lack of Adaptivity: Frequency of accesses is not considered, so no locality is considered in this algorithm.

2 LRU Page Replacement Policy

In this policy, we can do the same thing as what was done for FIFO, the only difference being: Each time a page is accessed and is in memory, take it out of the queue and enqueue it into the back of the queue again. If it is not in memory, then enqueue it and, if required, evict the page at the head of the queue.

2.1 Running the code

Run the bash script provided in the following format:
`bash run_lru.sh <arg1> <arg2> <arg3> <arg4>` , where the arguments are exactly as described in the problem statement. (arg1 is number of addressable pages in the system, arg2 is number of pages/frames in main memory, arg3 is number of pages in swap space of disk, and arg4 is input file name)

2.2 Advantages

- Good performance: This approach tries to minimize number of page faults by evicting the page in memory that was previously accessed furthest in the past.
- High Adaptivity: Temporal Locality is a strong consideration taken by this algorithm. So, this scales well for systems where accesses to pages often repeat within a time frame.

2.3 Limitations

- Overhead costs: This tends to have more overhead in terms of computation as it requires maintaining a list of recently used pages and updating it every time a page is accessed. Maintaining timestamps would be too costly, so often an approximate approach is taken.
- Quick Locality Shifts: If temporal locality changes quickly, then LRU would fail to minimize page faults in an optimal way
- Small Working Set: If the working set (i.e., the set of pages accessed frequently) is small, then LRU policy may not perform optimally. This is because it may evict pages that are still likely to be accessed in the near future.

3 Random Page Replacement Policy

In this policy, a random page is picked each time for a page to be stored each time it arrives, and if a page is already there, then it is evicted. This could be seen as being ‘brutally fair’.

3.1 Running the code

Run the bash script provided in the following format:
`bash run_random.sh <arg1> <arg2> <arg3> <arg4>` , where the arguments are exactly as described in the problem statement. (arg1 is number of addressable pages in the system, arg2 is number of pages/frames in main memory, arg3 is number of pages in swap space of disk, and arg4 is input file name)

3.2 Advantages

- Fairness: This policy treats all pages in memory equally, and has no bias towards not evicting a certain page.
- Simple implementation: No need for bookkeeping is present here, since evictions are at random.
- No corner cases: This approach doesn’t go into any corner cases due to the element of randomness.

3.3 Limitations

- Unpredictable and irregular: Since it is random, it may get unlucky and have many consecutive instances of unlucky evictions.
- Lack of Adaptivity: Frequency of accesses is not considered, so no locality is considered in this algorithm.

4 Graphical Representation

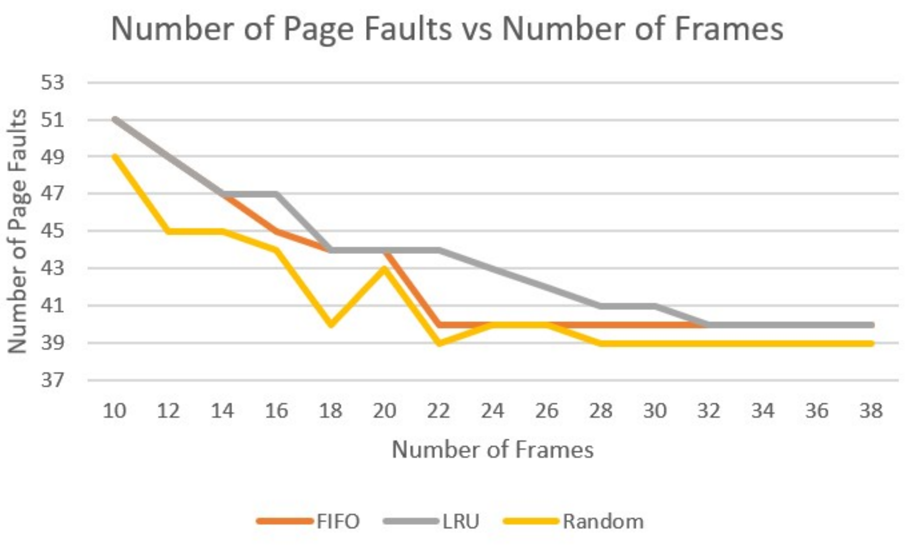

Figure 1: For req1.dat

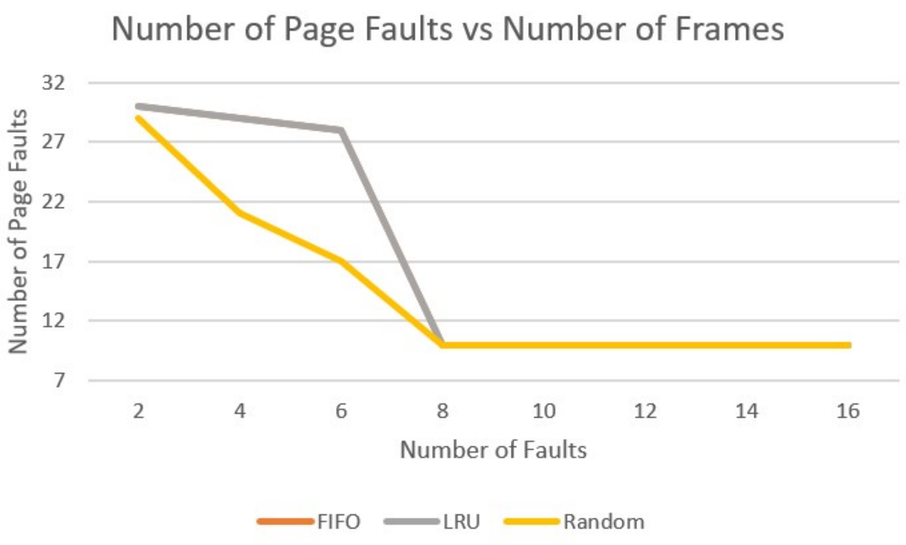

Figure 2: For req2.dat

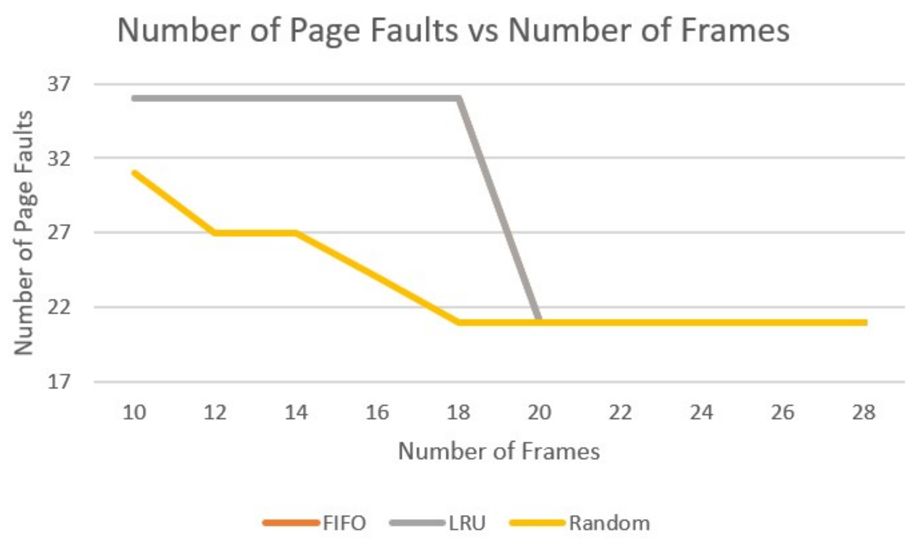

Figure 3: For req3.dat

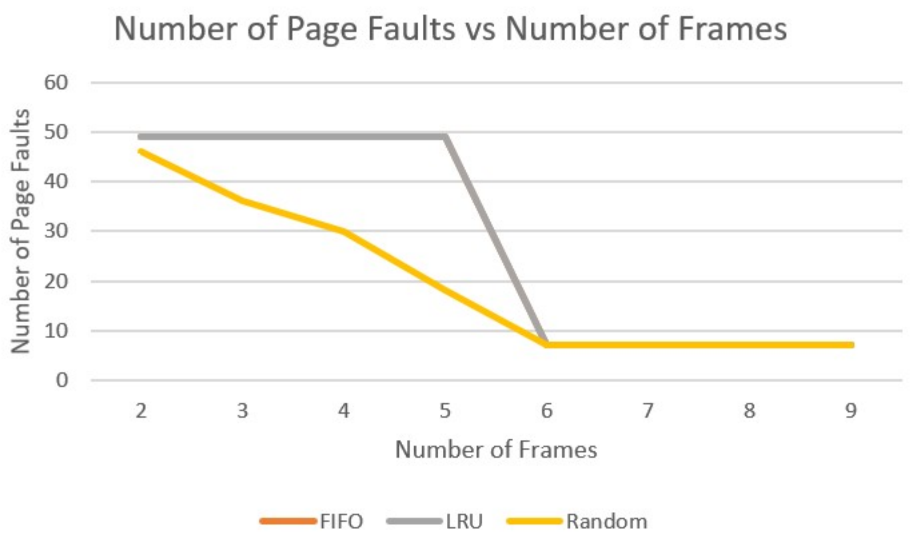

Figure 4: For req4.dat

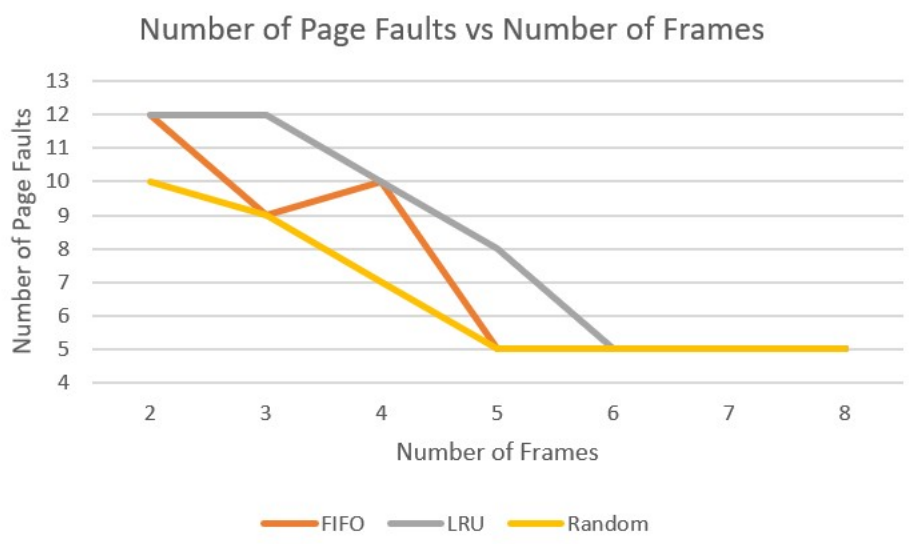

Figure 5: For req5.dat

5 Points to be noted

- In three of the graphs, LRU and FIFO are overlapping. This is due to the possibility of the least recently accessed also being the oldest in terms of first access.
- Random tends to have some degree of irregularity, which is in its intrinsic nature.
- The fifth file demonstrates Belady's anomaly. This is a phenomenon that occurs in page replacement algorithms where increasing the number of page frames available to the system can result in more page faults. In other words, adding more memory to a system can cause it to perform worse, which is counter-intuitive.
- Eventually, there is observed to be stagnation in the number of page faults when space in memory is enough, to the point where the misses are due to cold (compulsory) misses.