Majd & Mohamad Soufan

4/7/2017

Dr. Hwang

CS470

Memory Management Project Discussion

In this project, we created a memory management unit class that supports all three algorithms. The project is supported with a friendly-user GUI that is very easy to interact with. We also created three threads that run all three algorithms concurrently, which made the program run much faster on all the test files. The FIFO algorithm used the queues data structure because they are very easy to interact with. Replacing pages is determined by FIFO. So, the first page to come in is the first page to go out or to be replaced. There are no specific procedures that must be taken if the coming page does not exist in the frames. The program picks the first page in the frames' queue and pop it. For LRU, we used the List data structure that holds PPreferences objects. The PPreferences is a class that we created that hold the page references with their process ID. Our program save all the page references in a list and creates the frames List in the LRU function. If the frames are not filled yet, every new page is a fault page. When the frames are filled, the LRU checks if the new page reference exist in the frames list. If it does, then skip this page references. If it does not, then look back into the page references' list and check for the indexes of each page in the frames' list. Pick the least one and replace the page reference in the frames list with the new page reference. OPT strategy is more difficult to implement than FIFO. OPT requires the program to keep track of order of the pages in the past and anticipating/realizing the ones coming in the future. In my implementation, I used a list of

PPReferences to keep track of the pages that are coming, and I also used a Virtual Memory List (representing frames) that will be update after each entry/page. So, if the page is already in the list it will be moved to the end of the list, or if it is new page it will go immediately to the end of the list. In this way, the VMList is always in order from older to most recent. Whenever I get a new page and all the frames are taken the page replacement algorithm will go and learn which page is going to be the last to be referenced in the reference string (i.e. PPReference list) or is not referenced anymore and it will replace that page with coming page. In case there are more than one page that will not be referenced any more the algorithm will pick the older one for the replacement.

Q&A (Majd)

1. The OPT algorithm results are impressing. The number of page faults are very minimal in comparison to the other algorithms. However, it's runtime is longer than FIFO and LRU.
On the other hand, FIFO is the exact opposite. It is the fastest algorithm, and the easiest to implement, but it shows more fault pages. LRU is very efficient. It does provide fewer pages than FIFO, but it's run time is not as fast. With regard to the number of processes and s & o percentages the results for LRU and FIFO and close with a little advantage for LRU, however, OPT is better than both generally, and especially for fewer number of frames

2. The OPT was the most difficult algorithm to implement because it looks into the next page references, and if it does not find them it looks back to check which one has been in the frames for the most time.

3. FIFO. This algorithm uses the queue and it does not check for next nor the previous pages

4. Nothing

5. This concept was very similar to one of the concept that I took in the Networks class. I think the OPT was the most surprising because I have never used such an algorithm before.

Q&A (Mohamad)

1. Picking a single strategy is a hard task because it depends on several variables. For example, FIFO is very easy to implement and it is fast to execute, however, it results with the most number of page faults. LRU, is more difficult to implement but it has better results than FIFO, especially for small number of frames. It is not as fast though. OPT is the hardest to implement, however, it results with the fewest number of page faults. Number of processes and the percentages of s and o forms a great variant between LRU, FIFO and OPT. OPT shows much better result, when LRU is just a little bit better than FIFO. This statement is more true when fewer number of frames is in the virtual memory
2. Some of the strategy requires keeping track of history and future for page references.
3. Implementing FIFO
4. Learn more about queues and utilize them in OPT if possible
5. Data structures used in memory can change in size which is very useful but also requires more cautious when implementing it.