AOS Project

Page Replacement Algorithms

Aakash Tripathi (2022201053)
Harsimran Singh (2022201049)
Jatin Sharma (2022201023)
Nikhil Chawla (2022201045)

https://github.com/akash18tripathi/Page-replacement-policies-simulation

The purpose of this project is to write a memory management simulator that uses paging and measure the performances of page replacement algorithms.

FIFO

In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue.
When a page needs to be replaced page in the front of the queue is selected for removal.

Pros

• Low-overhead implementation

Cons

• May replace the heavily used pages

FIFO with second chance

Whenever a Page is accessed again, its reference bit is set to 1. Now when this page comes contender for replacement , it will be given second chance
by setting its reference bit to 0 and pushing it to end of queue.

Pros

• Simple to implement

Cons

• The worst case may take a long time

Clock

In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin matter, and
a page that has been accessed between consecutive considerations will not be replaced.
The page replaced is the one that, when considered in a round robin matter, has not been accessed since its last consideration.
It is implemented by adding a “second chance” bit to each memory frame-every time the frame is considered (due to a reference made to the page inside it), this bit is set to 1, which gives the page a second chance, as when we consider the candidate page for replacement, we replace the first one with this bit set to 0 (while zeroing out bits of the other pages we see in the process). Thus, a page with the “second chance” bit set to 1 is never replaced during the first consideration and will only be replaced if all the other pages deserve a second chance too.

Pros

• Simple to implement

Cons

• The worst case may take a long time

Least Recently Used

The LRU policy swaps out the pages which have been referred least recently out of the memory in case if no frames are free to accommodate any pages. This is achieved by

maintaining all the pages in a linked list. The pages which are referred are instantly removed from the list and added at the end of it.
In this way, as you go from left end of the linked list to the right, you will encounter pages which are referred more recently. In case if memory has no frames left to accommodate any new pages, the page which is at the left end of the list is removed from the frame (and also from the list). The new page is added at the emptied frame (and to the right end of the list).
Hits and misses are found depending upon the existence of the page in the list. If the page is present in the linked list, then it is a hit, else it is a miss.

Pros

- Remembers history of pages.
- Free from Belady Anomaly.

Cons

- It requires no hardware support

Not Frequently Used

The NFU policy swaps out the pages which have been least frequently used from the memory in case if no frames are free to accommodate any new page. In case if two pages have same frequency of reference, then the page which is least recently referred is removed. This is achieved by maintaining two maps. One map maps the pages to their frequency of reference and the other maps a page to its address in LRU list.
In case when a page has to me removed from the memory, the page which has the lowest counter is found using the frequency map. There can be many pages which can be mapped to the same frequency. In this case, the page which is least recently referred is chosen.

Pros

- Free from Belady Anomaly.
- Takes history into consideration.

Cons

- If a page is heavily used, its counter becomes very large.
- If program’s behavior changes and the largest referred page is no longer used, then the most referred page’s counter never decreases and hence, it will never be chosen for replacement.

Aging

The aging policy assumes that the page which is the oldest is most suitable for replacement. Every page which is in the memory has a counter associated with it. The page which is most recent is given a very large counter which indicates that it is the youngest page in the memory. With every clock cycle, the counter decreases by 1, which indicates its increasing age.
Now in case if a page has to be replaced, then the page having the least counter will be chosen. If an already existing page is referred, then its counter will be set again to the max_value.

Pros

- Free from Belady Anomaly.

Cons

- Assumes older pages will be referenced less frequently in future.

Optimal page replacement algorithm

Target is to minimize number of page faults. In this algorithm, OS replaces the page that will not be used for the longest period of time in future.

For example, a page that is not going to be used for the next 5 seconds will be swapped out over a page that is going to be used within the next 0.5 seconds.

Optimal page replacement algorithm is not practical as we cannot predict future. However it is used as a reference for other page replacement algorithms.

Pros

- Best hit ratio.

Cons

- Practically near to impossible to implement.

Random page replacement algorithm

Random replacement algorithm replaces a random page in memory. This eliminates the overhead cost of tracking page references. Usually it fares better than FIFO, and for looping memory references it is better than LRU, although generally LRU performs better in practice.

Efficiency of randomized algorithm for the paging problem is measured using amortized analysis.

Not Recently Used

The not recently used (NRU) page replacement algorithm is an algorithm that favours keeping pages in memory that have been recently used. This algorithm works on the following principle: when a page is referenced, a referenced bit is set for that page, marking it as referenced. Similarly, when a page is modified, a modified bit is set.

At a certain fixed time interval, a timer interrupt triggers and clears the referenced bit of all the pages, so only pages referenced within the current timer interval are marked with a referenced bit.

When a page needs to be replaced, the operating system divides the pages into four classes:

3. referenced, modified
2. referenced, not modified
1. not referenced, modified
0. not referenced, not modified

Although it does not seem possible for a page to be modified yet not referenced, this happens when a class 3 page has its referenced bit cleared by the timer interrupt. The NRU algorithm picks a random page from the lowest category for removal. So out of the above four page categories, the NRU algorithm will replace a not-referenced, not-modified page if such a page exists.

This algorithm implies that a modified but not-referenced (within the last timer interval) page is less important than a not-modified page that is intensely referenced.

Working Set

Input is max number of frames(window size) . The number of frames is dynamic in working set. The number of frames at any instance is equal to the distinct pages present in that window at that instance

Working Set Clock

It is similar to clock, but it has one more parameter as threshold, and time is noted for each reference page. When we remove a page, we check if value is > threshold, if yes then page is removed.

Comparing these algorithms

- For each of these algorithms, following are provided -
  - number of frames
  - vector of int contains page numbers
- Performance of these algorithms is compared in terms of hit ratio.
- There are three parameters,
  - vector containing page numbers
  - min_frames
  - max_frames
- For the given sequence of page numbers, hit ratio is found for a x number of frames where x ranges between min_frames and max_frames.
- Then this hit ratio is plotted for every x.
- Run command: `python3 driver.py`