Andy Qin

COEN177 L

01 March 2022

COEN 177 Lab 8 Report

Objective

* Report on implementation of common page replacement algorithms, explain steps with examples, and elaborate on advantages and disadvantages of each.
* Provide collected data and results and analyze them to make conclusions.

Procedures

1. **Implementation of Page Replacement Algorithms**

**First-In-First-Out Page Replacement (FIFO)**

Text

Description automatically generated

**Logic**: As the name of the algorithm suggests, the first-arriving page (the oldest page) will be replaced first as well. If page already exists, move to the next page.

**Steps:**

* We first check if the requested page is in the page table already; if yes, we do not do anything to the page table; simply update the age of the requested page (like replacing it with new ones) and go to the next oldest page.
* If we do not find the page, we replace the page number at current position with the newly requested page, and move forward to the next oldest page position
* The next oldest is simply the next position of the page table as a circular queue is used. The current position updates every page lookup.

**Second-Chance Page Replacement**

Text

Description automatically generatedText

Description automatically generated

**Logic**: Second-Chance Algorithm is an improvement based on FIFO, but it gives every page a second chance. The algorithm still goes in a circular loop.

**Steps**

* A new Boolean variable “index” is added to the struct “ref\_page” to keep track of a page’s second chance.
* If page already exist in the table, set “second\_chance” to “true”. We can think about it has giving a page two lives --- if a page request misses, the page at the current position will loses a life; if it only has one life, it will get replaced; if it has two lives, it loses one life by setting the “second\_chance” to “false”. Move the current position forward.
* Also move the position forward for every page lookup.

**Least-Recent-Used (LRU)**

Text

Description automatically generatedText

Description automatically generated

Logic: A new page will be replaced by a least recently used page. It makes sense as it has a lesser chance to be visited again soon compared to the other pages.

Steps:

* We keep track of the time that a page has been in the able, use a int variable called “index’ to store the value.
* If page request found in table, refresh the time to 0 (index = 0); every other page increases their time in the table.
* If page not found, replace the page that has the largest time in the table (the largest index).

1. **Testing of performance of each algorithm**

**Custom textinput.txt**

* 20 random page requests generated using textinput.c

Command Line Test:

* Cat textInput.txt | ./fifo 5
* Cat textInput.txt | ./sec\_chance 5
* Cat textInput.txt | ./lru 5
* Cat 🡪 use textInput.txt as input
* ./ --. Runs the executable page replacement algorithms
* 5 🡪 number of slots in the page table

FIFO

Graphical user interface, text

Description automatically generated

Second-Chance

Text

Description automatically generated

LRU

Graphical user interface, text

Description automatically generated

**Accesses.txt --- test provided by Lab**

* **Consisted of 10k predetermined page requests**

Command Line Test:

* Cat textInput10k.txt | ./fifo 50
* Cat textInput10k.txt | ./sec\_chance 50
* Cat textInput10k.txt | ./lru 50
* Cat 🡪 use textInput.txt as input
* ./ --. Runs the executable page replacement algorithms
* 50 🡪 number of slots in the page table

FIFO

A black background with white text

Description automatically generated with low confidence

Second-Chance

A black background with white text

Description automatically generated with low confidence

LRU

A black screen with white text

Description automatically generated with low confidence

Realize that we are using small page requests and small page able size in the beginning. The number of page requests is way bigger than this, and the page table can contain more slots. This is used mainly for checking if algorithms are implemented correctly.

The test with accesses.txt is a more common theme in reality as there are usually thousands of page requests coming in at the same time.

The result I got matches with the official test results, which means I can safely move to the next step.

1. **Table and Charts**

**More Testing --- using test.sh**

Text

Description automatically generated**Text

Description automatically generated**

Text

Description automatically generated

FIFO

|  |  |  |
| --- | --- | --- |
| Table Size | Page Faults | Hit Rate |
| 10 | 9916 | 0.0084 |
| 50 | 9515 | 0.0485 |
| 100 | 9018 | 0.0982 |
| 250 | 7534 | 0.2466 |
| 500 | 5130 | 0.487 |

Second-Chance

|  |  |  |
| --- | --- | --- |
| Table Size | Page Faults | Hit Rate |
| 10 | 9915 | 0.0085 |
| 50 | 9510 | 0.049 |
| 100 | 9029 | 0.0978 |
| 250 | 7532 | 0.2474 |
| 500 | 5206 | 0.4822 |

LRU

|  |  |  |
| --- | --- | --- |
| Table Size | Page Faults | Hit Rate |
| 10 | 9915 | 0.0085 |
| 50 | 9510 | 0.049 |
| 100 | 9022 | 0.0971 |
| 250 | 7526 | 0.2468 |
| 500 | 5178 | 0.4794 |

Conclusion

* The three page replacement algorithms have really similar performances in hit rates with different page table sizes.
* Second-Chance has just slightly better performance in each stages except for page size = 500