Project Report

Course Title : Operating Systems

Course Code : CSE325

Semester : Fall 2023

Section : 05

Submitted by:

Junnun Mohamed Karim (2022-1-60-108)

Submitted to:

Md. Mahir Ashhab

Lecturer

Department of Computer Science and Engineering

Submission Date:

23rd December, 2023

Introduction

In the domain of memory management algorithms, the realization of the Least Recently Used (LRU) algorithm, although theoretically feasible, encounters challenges due to hardware limitations on many machines. Consequently, a demand arises for a software-based solution, leading to the emergence of the Not Frequently Used (NFU) algorithm. NFU incorporates a software counter associated with each page, initialized to zero. At regular intervals, typically synchronized with clock interrupts, the operating system assesses the status of all pages in memory. The algorithm integrates the Reference (R) bit, reflecting whether a page has been referenced, into the counter. These counters effectively approximate the frequency of reference for each page, with the page possessing the lowest counter value chosen for replacement in the event of a page fault.

However, NFU exhibits a noteworthy limitation—it tends to "never forget,". In scenarios such as multipass compilers, where certain pages are heavily used in the initial passes, NFU may retain high counts well into subsequent passes. This persistence leads to a predicament where the operating system may prematurely eliminate pages still in active use while retaining those from earlier phases.

Fortunately, a modification known as aging enhances NFU's performance, allowing it to simulate LRU more effectively. This modification involves right-shifting each counter by one bit before incorporating the R bit and placing the R bit at the leftmost position. The resulting algorithm, called aging, addresses the longevity issue of NFU, introducing a finite past horizon for counters and enhancing the algorithm's ability to differentiate between recent and older references.

The aging algorithm represents a refined version of NFU. By shifting counters and strategically placing the R bit, it provides a nuanced perspective on page reference frequency. When a page fault occurs, the algorithm replaces the page with the lowest counter value, factoring in the modified aging mechanism.

Implementation Details

The NFU algorithm is implemented in python. Here's an overview of the key components and functionalities:

Data Structure:

- Python List was used to store page and frame informations because of its ease of use and built-in functions.

Page Representation and Initialization:

- The program begins by defining constants such as `NUM_PAGES` to 16, `NUM_FRAMES` to 8, and `REFERENCE_COUNTER` to 8 to configure the simulation parameters.

    - `NUM_PAGES` is used to set the number of pages to divide the virtual memory.

    - `NUM_FRAMES` is used to set the number of page frames the memory will have.

    - `REFERENCE_COUNTER` is used to keep track of the aging time of each page

- A function named `page_shift_ref_bit` is implemented to shift bits to the right within the reference counters associated with each page.

Page Reference Counting:

- The function `page_ref_count` is responsible for calculating the reference count for each page based on the presence of the Reference (R) bit.

- The script maintains a list `page_ref_count_list` to store the calculated counts for each page.

Page Fault Handling:

- The `page_fault` function is invoked when a page fault occurs. It identifies the page to be replaced based on the lowest reference count and updates the page reference counters accordingly.

Main Simulation Loop

- The `main` function handles the simulation, utilizing a clock tick mechanism to advance the simulation time.

- Random page reference patterns are generated within a loop, simulating the dynamic nature of page references over time.

- Page tables, frame contents, clock ticks, and replacement counts are logged to an output file, providing a detailed view of the simulation process.

Simulation Execution:

- The program concludes with the execution of the simulation through the `main` function. The output is directed to a file named "output.txt."

Note:

- The program employs a sleep interval of 0.001 seconds between iterations to simulate the passage of time.

This implementation offers a practical demonstration of the NFU algorithm's behavior in managing page references within a simulated environment. The dynamic nature of page references, along with the associated counters and frame management, allows for a comprehensive exploration of the algorithm's efficacy in handling page faults and optimizing memory usage.

Trace Explanation

First 8 clock ticks of the trace file is explained below:

Clock Tick 0:

- Initialization:
  - Reference counters for all pages are set to 0.
  - Page table and frame table are initialized.
- Page references:
  - Pages 0, 3, 4, 7, 8, and 10 are referenced (indicated by 1s in the page table).
- Page faults and replacement:
  - All referenced pages cause page faults as the frames are empty.
  - Pages 0, 3, 4, 7, 8, and 10 are inserted into empty frames.
- Page table and frame table:
  - Page table shows which pages are present in memory (1) or not (0).
  - Frame table shows the page index stored in each frame.

Clock Tick 1:

- Page references:
  - Pages 1, 2, 5, 9, 12, and 15 are referenced.
- Page faults and replacement:
  - No page faults occur as all referenced pages are already in memory.
  - Reference counters for the referenced pages are updated.

Clock Tick 2:

- Page references:
  - Pages 11 and 13 are referenced.
- Page faults and replacement:
  - Pages 11 and 13 cause page faults as there are no empty frames.
  - The NFU algorithm selects page 0 (with the lowest reference counter) for replacement.
  - Pages 11 and 13 are inserted into the frames previously holding pages 0 and 8.

Clock Tick 3:

- Page references:
  - Pages 1, 4, 5, 9, 12, and 15 are referenced.
- Page faults and replacement:
  - Pages 4 and 15 cause page faults.
  - The NFU algorithm selects pages 7 and 8 for replacement (having lower reference counters than other pages).
  - Pages 4 and 15 are inserted into the frames previously holding pages 7 and 8.

Clock Tick 4:

- Page references:
  - No pages are referenced.
- Page faults and replacement:
  - No page faults occur.
  - Reference counters remain unchanged.

Clock Tick 5:

- Page references:
  - Pages 2, 5, 9, and 15 are referenced.
- Page faults and replacement:
  - No page faults occur.
  - Reference counters for the referenced pages are updated.

Clock Tick 6:

- Page references:
  - Pages 1, 4, 5, 6, 9, 10, 11, and 15 are referenced.
- Page faults and replacement:
  - Page 4 causes a page fault.
  - The NFU algorithm selects page 15 for replacement.
  - Page 4 is inserted into the frame previously holding page 15.
  - However, page 15 is immediately referenced again, causing another page fault.
  - The NFU algorithm again selects page 4 for replacement (as it has the lowest reference counter).
  - Page 15 is inserted back into the frame previously holding page 4.

Clock Tick 7:

- Page references:
  - Pages 3, 6, and 15 are referenced.
- Page faults and replacement:
  - Page 3 causes a page fault.
  - The NFU algorithm selects page 15 for replacement.
  - Page 3 is inserted into the frame previously holding page 15.
  - Page 6 causes another page fault.
  - The NFU algorithm selects page 4 for replacement.
  - Page 6 is inserted into the frame previously holding page 4.
  - Page 15 is referenced again, causing another page fault.
  - The NFU algorithm selects page 3 for replacement.
  - Page 15 is inserted back into the frame previously holding page 3.