Advanced Page Replacement Algorithms

1 Introduction to Paging

Paging is a memory management scheme used by operating systems to handle memory allocation efficiently. It allows the physical memory (RAM) to be divided into fixed-size blocks called **frames**, while the logical memory (used by processes) is divided into blocks of the same size called **pages**. This abstraction enables the operating system to manage memory more effectively, especially in systems with limited physical memory.

1.1 Key Concepts of Paging

- **Pages and Frames:**
  - Pages are fixed-size blocks of logical memory used by processes.
  - Frames are fixed-size blocks of physical memory in RAM.
- **Page Table:**
  - A data structure used by the operating system to map logical addresses (pages) to physical addresses (frames).
  - Each entry in the page table contains the frame number corresponding to a page.
- **Page Fault:**
  - Occurs when a process attempts to access a page that is not currently in physical memory.
  - The operating system must handle the page fault by loading the required page into memory, potentially evicting another page.
- **Virtual Memory:**
  - A technique that allows processes to use more memory than physically available by swapping pages between RAM and secondary storage (e.g., disk).

1.2 Importance of Page Replacement Algorithms

When a page fault occurs, the operating system must decide which page to evict from memory to make room for the new page. This decision is critical for system performance, as inefficient page replacement can lead to excessive page faults, known as **thrashing**. Page replacement algorithms aim to minimize page faults and optimize memory usage.

2 Traditional Page Replacement Algorithms

Before diving into advanced algorithms, it is essential to understand traditional page replacement algorithms, which form the foundation for more sophisticated approaches.

2.1 FIFO (First-In-First-Out)

- **Concept:** The oldest page in memory is evicted first.
- **Advantages:** Simple to implement.
- **Disadvantages:** Does not consider the frequency or recency of page usage, leading to poor performance in many cases.

2.2 LRU (Least Recently Used)

- **Concept:** The page that has not been used for the longest time is evicted.
- **Advantages:** Better performance than FIFO, as it considers recency of use.
- **Disadvantages:** Requires significant overhead to track page usage.

2.3 Clock Algorithm

- **Concept:** A circular list of pages is maintained, and a "clock hand" scans the list. Pages with a reference bit of 0 are evicted.
- **Advantages:** More efficient than LRU, with lower overhead.
- **Disadvantages:** Does not distinguish between frequently and infrequently used pages.

3 Advanced Page Replacement Algorithms

Traditional algorithms have limitations in handling complex memory access patterns. Advanced algorithms like **Clock-Pro** and **WSClock** address these limitations by incorporating more sophisticated mechanisms.

4 Clock-Pro Algorithm

The Clock-Pro algorithm is an enhancement of the basic Clock algorithm, designed to approximate the LRU policy while being computationally efficient. It uses a circular list of pages, similar to the Clock algorithm, but introduces additional pointers and structures to better handle page replacement. Clock-Pro is particularly effective in systems with large memory sizes and complex access patterns.

4.1 Key Features

- **Circular List:** Pages are organized in a circular list, each with a reference bit indicating recent access.
- **Two Hands:** The algorithm maintains two pointers:
  - **Cold Hand:** Scans for cold pages (infrequently accessed) to evict.
  - **Hot Hand:** Ensures hot pages (frequently accessed) are not evicted prematurely.
- **Page Classification:** Pages are classified as either **hot** (frequently accessed) or **cold** (infrequently accessed).
- **Efficient Eviction:** Cold pages are evicted first, while hot pages are given more opportunities to remain in memory.

4.2 How it Works

1. When a page is accessed, its reference bit is set to 1.
2. The cold hand scans the circular list, looking for pages with a reference bit of 0 (cold pages).
3. If a page has a reference bit of 1, it is reset to 0, and the hand moves forward.
4. If a page is cold (reference bit = 0), it is evicted.
5. The hot hand ensures that hot pages are not evicted prematurely by protecting them during the scanning process.

5 WSClock Algorithm

The WSClock (Working Set Clock) algorithm combines the concepts of the Clock algorithm and the working set model. It aims to keep the working set of each process in memory to minimize page faults. The working set of a process is the set of pages that have been accessed within a specified time window (τ).

5.1 Key Features

- **Circular List:** Pages are organized in a circular list, each with a reference bit and a time-of-last-use field.
- **Time Window (τ):** A parameter that defines the working set. Pages not accessed within this window are considered for eviction.
- **Reference Bit and Time-of-Last-Use:** Each page tracks whether it has been recently accessed and the time of its last access.
- **Eviction Policy:** Pages outside the working set (i.e., not accessed within the time window) are evicted.

5.2 How it Works

1. When a page is accessed, its reference bit is set to 1, and its time-of-last-use is updated.
2. The algorithm scans the circular list, looking for pages with a reference bit of 0.
3. If a page has a reference bit of 1, it is reset to 0, and the hand moves forward.
4. If a page has a reference bit of 0 and its time-of-last-use is outside the time window (τ), it is evicted.
5. If no such page is found, the algorithm evicts the page at the current hand position.

6 Numerical Questions on Clock-Pro and WSClock

6.1 Clock-Pro Questions

1. **Problem:**
   - Consider a system using the Clock-Pro algorithm with the following page reference sequence: $1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5$.
   - Assume the memory can hold 3 pages.
   - Simulate the Clock-Pro algorithm and determine the number of page faults.
2. **Problem:**
   - A system uses the Clock-Pro algorithm with 4 frames. The reference string is: $7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0$
   - Simulate the algorithm and calculate the total number of page faults.
3. **Problem:**
   - Given a memory with 5 frames and the reference string: $1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$.
   - Use the Clock-Pro algorithm to determine the number of page faults.

6.2 WSClock Questions

1. **Problem:**

   - A system uses the WSClock algorithm with a time window (τ) of 4 units. The reference string and access times are:

     ```
     Page: 1, 2, 3, 4, 1, 2, 5
     Time: 1, 2, 3, 4, 5, 6, 7
     ```

   - Assume the memory can hold 3 pages. Simulate the WSClock algorithm and determine the number of page faults.

2. **Problem:**

   - A system uses the WSClock algorithm with a time window (τ) of 5 units. The reference string and access times are:

     ```
     Page: 1, 3, 2, 4, 1, 3, 5, 2, 4, 5
     Time: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
     ```

   - Assume the memory can hold 4 pages. Simulate the algorithm and calculate the total number of page faults.

3. **Problem:**

   - Given a memory with 5 frames and a time window (τ) of 3 units, use the WSClock algorithm to determine the number of page faults for the following reference string:

     ```
     Page: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
     Time: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
     ```

7 Solutions to Selected Problems

7.1 Clock-Pro Example

- **Reference String:** $1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5$
- **Memory Frames:** 3
- **Page Faults:** 9

7.2 WSClock Example

- **Reference String:** $1, 2, 3, 4, 1, 2, 5$
- **Time Window (τ):** 4
- **Memory Frames:** 3
- **Page Faults:** 5

8 Key Observations

1. **Clock-Pro:**

   - Pages are classified as hot or cold based on their reference bits.
   - The cold hand evicts cold pages, while the hot hand ensures hot pages are not evicted prematurely.
   - Resetting reference bits helps approximate LRU behavior.

2. **WSClock:**
    - Pages are evicted if they are outside the working set (i.e., not accessed within the time window τ).
    - The algorithm combines the Clock algorithm's efficiency with the working set model's focus on recent usage.

9 Conclusion

Page replacement algorithms are essential for efficient memory management in operating systems. While traditional algorithms like FIFO and LRU provide basic functionality, advanced algorithms like Clock-Pro and WSClock offer improved performance by incorporating sophisticated mechanisms to approximate optimal page replacement behavior. Understanding these algorithms is crucial for designing and optimizing modern operating systems.