FIFO and LRU Implementation Report

Aman Hogan-Bailey

2238-CSE-5331-002, Professor Ashraf Aboulnaga

The University of Texas at Arlington

Contents

Overall Status

The main components implemented were the page reference count, BHR variable tracking, and LRU and FIFO replacement policies. The overall project status and implementation are complete. There are three things of note:

- Upon printing the top k pages, some load counts for these pages will differ from the sample output by one; however, this did not seem to change the order of the top k pages concerning the hit count and did not appear to be a logical error.
- The `GlobalConst.java` did not work at times, so I defined local constants in the respective files.
- FIFO and LRU will return a `Buffer pool exceeded exception` on `BHR MRU TEST` for buffer size `15`, which I deduce is the expected output since all pages are pinned, the buffer pool is small, and FIFO and LRU replacement is not optimal for that situation.

Below is the step-by-step process of how the project components were completed and implemented:

1. Reviewed the functionality of the Buffer Manager
2. Focused on the `pinPage()` functionality.
3. Researched and implemented the FIFO and LRU page replacement algorithms.
4. Implemented the Clock algorithm and tried to see if BHR calculations were in line with the sample output.
5. Modified and debugged the buffer manager for LRU and FIFO specifically to get the correct BHR variables and counts.
6. Reviewed the code and refined documentation across the edited files.

File Descriptions

List of Files

All the code included in the assignment was used in the buffer manager implementation. Below is a list that details the main source files and classes that guide the buffer manager and replacement policy functionality:

- `BufMgr.java` - **MODIFIED**
- `Lru.java` - **CREATED**
- `FIFO.java` - **CREATED**

BufMgr.java

Here is an overview of the major functions and data structures modified or created in the `BufMgr.java` class.

- Data Structures
    - 2D array for page references
- Main Functions
    - Removed `uniquePageLoads`, `pageLoadHits`, and `page-fault` calculation functionality because it was nonessential.
    - Added an array to keep track of the page reference array that keeps track of pid, loads, hits, and the number of times that page was a victim.
    - Calculated BHR, page hits, page load requests, and the individual page BHR.
    - Kept track of page hits, page load requests, and page specific information in the page reference array each time the `pinPage()` function was called.
    - Sorted, printed, and logged to an output file the details contained in the page reference array.

FIFO.java

Here is an overview of the major functions and data structures created in the `FIFO.java` class.

- Data Structures
    - Queue
- Main Functions
    - `pickVictim()`:
        - Created `nextFrameToReplace` variable to iterate through the frames in a circular manner.
        - If a frame is found with the `PINNED` state, it continues to the next frame until an available frame is found or it completes one full cycle.
        - If all frames are pinned, it returns `-1` indicating that there is no available frame. Otherwise, it returns the index of the selected victim frame.

Lru.java

Here is an overview of the major functions and data structures created in the `Lru.java` class.

- Data Structures
    - Array
- Main Functions
    - `update():`
        - Update the order of frames in the frames array when a frame is accessed.
        - Check if the frame is already in the array and updates its position.
        - The idea is to move the most recently used frame to the end of the array (MRU) and the least recently used frame to the beginning (LRU).
    - `pickVictim():`
        - If there are available frames (less than the buffer size), it selects the next available frame, marks it as `PINNED`, and returns its index.
        - If there are no available frames, it iterates through the frames in the order specified by the frames array.
        - It selects the first unpinned frame it encounters, marks it as `PINNED`, updates the order of frames using the update method, and returns its index.
        - If no frames are available (all are pinned), it returns -1 indicating that there is no available frame.

There were no additional test cases for the LRU and FIFO implementations other than the assigned `15` and `42` buffer sizes.

Division of Labor

I worked alone so there was no division of labor. In total, the assignment took about 40 hours to complete, with at least 90% of that time comprising of researching or debugging.

Logical Errors

Below are points that detail experiences and challenges encountered during the implementation and debugging of the buffer manager and page replacement policies (specifically FIFO and LRU):

- **Using `tempfd` instead of `pageno`:** I initially used the variable `tempfd` instead of `pageno` when checking the Page ID to determine if it's greater than 8. This caused discrepancies among the number of page loads and hits in the `pagerefcount` array.
- **Where to increment and count `pageLoadRequests` and `totalPageHits`:** I could not determine the correct locations in the code to increment counters for page load requests and total page hits. After trying the `unpinPage()` and `freePage(),` the `pinPage()` function made the most sense for incrementing these counts.
- **Keeping track of page details using the `pagerefcount` array:** I tried opting out of using a 2D array and instead tried using a `List` or an `ArrayList`, but the implementation did not work. For simplicity, I switched back to a 2D array implementation. I also decided to use the native `Java` sort function over the sort function given in the project.
- **`pickVictim()` function logic in both LRU and FIFO:** I kept getting issues when returning the current victim frame in both implementations. During the `BMTEST` runs, the `FIFO` and `LRU` crashed on TEST 2. I eventually figured out how to keep track of the number of frames, which was the original purpose of the `nframes` variable.

Policy Analysis with Respect to Hit Ratios

Below is a set of observations and hypotheses regarding the relationships between various factors and hit ratios in the context of FIFO and LRU replacement policies in the buffer management system. After performing BHR tests with a 15 and 42 buffer size, these are some general observations:

- The fewer page loads, the more concentrated the page hits are among the top K pages.
- The more page loads, the more evenly distributed page hits are among the Top K pages.
- The higher the BHR, the fewer number of loads per page among the Top K pages.
- The higher the buffer count, the fewer times a page becomes evicted among the Top K pages.
- The number of loads remains relatively evenly distributed among the Top K pages.

Buffer Pool Analysis

The following observations provide insights into the behavior of the FIFO and LRU page replacement policies under different buffer pool sizes and various test scenarios. After performing BHR tests with a 15 and 42 buffer size, these are some general observations:

- The Buffer Pool becomes full for the MRU TEST on buffer size of 15 but does not on elevated buffer sizes like 42. This observation indicates that the MRU TEST places a higher demand on the buffer pool, causing it to exceed its capacity when the buffer size is limited to 15.
- For the ROUND ROBIN TEST (A), the higher buffer size had a marginally better BHR than the smaller buffer sizes, indicating that a larger buffer can provide better performance in terms of reducing disk I/O for this specific scenario.
- For ROUND ROBIN TEST (B), the lower buffer size had a marginally better BHR than the greater buffer size. In Round Robin B, the opposite trend was observed. A smaller buffer size resulted in a slightly better BHR compared to a larger buffer size. This could be due to the specific access patterns or workload characteristics of the Round Robin B test.
- For the RANDOM TEST, the small buffer had a significantly higher BHR than the bigger buffer.

The observations regarding buffer size and performance, as well as the variations in different tests, were consistent across both the FIFO and LRU replacement policies, suggesting that the behavior of buffer pool management is influenced by buffer size and test scenarios rather than the specific replacement policy in this context. The optimal buffer size can vary depending on the specific requirements and constraints of the database system or application.

References

Figure 1 (FIFO table)

FIFO	
Test MRU	
15 Buffers	42 Buffers
Buffer Pool Exceeded	Aggregate Page Hits: 0
Buffer Pool Exceeded	Aggregate Page Loads: 3990
Buffer Pool Exceeded	Aggregate BHR (BHR1) : 0.0
Round Robbin (a)	
Aggregate Page Hits: 586	Aggregate Page Hits: 1686
Aggregate Page Loads: 225	Aggregate Page Loads: 630
Aggregate BHR (BHR1) : 2.60	Aggregate BHR (BHR1) : 2.68
Test Round Robin (b)	
Aggregate Page Hits: 729	Aggregate Page Hits: 1826
Aggregate Page Loads: 260	Aggregate Page Loads: 668
Aggregate BHR (BHR1) : 2.80	Aggregate BHR (BHR1) : 2.73
Test Random	
Aggregate Page Hits: 306	Aggregate Page Hits: 316
Aggregate Page Loads: 5	Aggregate Page Loads: 15
Aggregate BHR (BHR1) : 61.20	Aggregate BHR (BHR1) : 21.07

(Figure 1) FIFO output table for 15 buffers, 42 buffers, and for each BHR test.

Figure 2 (LRU table)

LRU	
Test MRU	
15 Buffers	42 Buffers
Buffer Pool Exceeded	Aggregate Page Hits: 0
Buffer Pool Exceeded	Aggregate Page Loads: 3990
Buffer Pool Exceeded	Aggregate BHR (BHR1) : 0.0
Round Robbin (a)	
Aggregate Page Hits: 586	Aggregate Page Hits: 1686
Aggregate Page Loads: 225	Aggregate Page Loads: 630
Aggregate BHR (BHR1) : 2.60	Aggregate BHR (BHR1) : 2.68
Test Round Robin (b)	
Aggregate Page Hits: 729	Aggregate Page Hits: 1826
Aggregate Page Loads: 260	Aggregate Page Loads: 668
Aggregate BHR (BHR1) : 2.80	Aggregate BHR (BHR1) : 2.73
Test Random	
Aggregate Page Hits: 306	Aggregate Page Hits: 316
Aggregate Page Loads: 5	Aggregate Page Loads: 15
Aggregate BHR (BHR1) : 61.20	Aggregate BHR (BHR1) : 21.07

(Figure 2) LRU output table for 15 buffers, 42 buffers, and for each BHR test.