CSc 360 Assignment #4

Due date: 11:59pm, Wednesday, 07 August 2013

Objectives

In this assignment, you will:

1. extend assignment 3 to include disk block caching;
2. implement the Least Recently Used (**LRU**) and Least Frequently Used (**LFU**) cache algorithms;
3. create test cases to evaluate/verify your approaches; and
4. complete a report describing your solution and tests.

Problem

This assignment is to be completed entirely in Java (no C coding). It differs from previous assignments in a few ways:

- you are given much more freedom over your approach to the assigned problems;
- testing is an important component;
- you must submit a 2-page report in pdf, doc or html format (no Doxygen, inline comments do not count).

These changes are reflected in the marking scheme, so be sure to review the *Evaluation* section.

Disk Cache [60%]

The goal of this assignment is to create a disk cache for the disk of assignment 3. A disk cache can allow us to avoid the large seek times we observed in assignment 3 by keeping a small cache of entries we have already viewed. Caching is one of the best ways to improve performance for a system. Many commerical operating systems use disk cache (or called buffer cache) heavily to reduce disk accesses.

Eventually as items are added to the cache, the cache fills up. When that happens an item in the cache must be replaced. Which item should be replaced? This problem leads to various solutions falling under *cache algorithms*. In this assignment we want you to complete two:

1. Least Recently Used (LRU) (30%): the least recently used cache item is discarded first.
2. Least Frequently Used (LFU) (30%) : the cache item that has been accessed the least in the past is discarded first.

The design and implementation of how you go about completing and testing these two algorithms is left to you.

Writing into a disk block when dealing with disk cache has its own set of issues. The simplest approach is to simply write immediately "through" into the disk. An alternative, and sometimes more effective approach, is to delay writing until the cache item is removed (i.e. mark it dirty first, and then write it "back" when replaced). The choice for which approach is left to you.

The empty files *DiskLRU.java* and *DiskLFU.java* file are provided for you to get started. They extend *DiskFCFS* for the sake of simplicity, but feel free to inherit to your previous solution of *DiskSSTF* (just make sure to include the *DiskSSTF.java* in your submission). In addition, two *VERY* simple test cases are provided to highlight the performance increases caching can provide. The bin files for them can be found in the *soln* folder.

Note: To keep things fair, the cache size should be 16.

Hint: Wondering where to start? Extend/Reimplement the *read()* function from *DiskScheduler*.

Report [20%]

This assignment requires you to write a report. In the report, you need to clearly explain your solution and testing.

Specifically, it must include

- a description of your approach including the data structures you used;
- a discussion of the performance bounds of different aspects of your approach;
- a discussion of the tests that you created, what they evaluate/demonstrate and their results.

The report should be approximately two pages. You may submit it in pdf, doc or html format. Feel free to include diagrams or graphs to help explain your solution.

Please put your report inside the *assign4/report* folder.

Evaluation

The assignment is to be done in teams of two. Each member will receive the same mark. Each team member should understand all aspects of the assignment. It is **HIGHLY** recommended that team members do the assignment together.

Your evaluation will be based mostly on the correctness of your solutions (60%). Testing is one way to evaluate your correctness and you will be evaluated on it (20%). Even if your code runs, there may still be errors which can only be uncovered by inspection. Obscured programming tricks and habits will make programs harder to understand. Keep everything simple and elegant!

The report counts for a significant part of the assignment (20%), so make sure to put some effort into it. The main evaluation for the report is how well you explain what you have done. Assume the audience for the report is fellow students.

NOTE: There is no Doxygen in this assignment. That said, please include some basic comments in

your Java code to assist the marker in quickly understanding your code.

- Code [60%]
- Testing [20%]
- Report [20%]

Submission

Most of your answers will be in the report, so please include references in your report to critical aspects of your code (helpful for the marker and you).

Please include all necessary makefiles in each directory to generate the executables and make sure that "make clean; make" will rebuild your solutions to each part in each subdirectories.

Submit a zipped file with the name *assignment4.tgz*.

assign4 should keep the structure of

- assign4
    - code
    - report

To do this go into cygwin and navigate to the path above assign4 and enter:

```
tar czvf assignment4.tgz assign4
```

Please submit this file (**assignment4.tgz**) electronically through Connex before midnight on the due date. **Only one team member needs to submit.**