Project 3: Virtual Memory Simulator[1]

Submit a gzipped tarball of your code to CourseWeb.

Due: Monday, November 12, 2018 @11:59pm
Late: Wednesday, November 14, 2018 @11:59pm with 10% reduction per late day

Table of Contents

[1] Based upon Project 3 of Dr. Misurda's CS 1550 course.

Project Overview

In class, we have been discussing various page replacement algorithms that an Operating System implementer may choose to use. In this project, you will compare the results of four different algorithms on traces of memory references. While simulating an algorithm, you will collect statistics about its performance such as the number of page faults that occur and the number of dirty frames that had to be written back to disk. When you are done with your program, you will write up your results and provide a graph that compares the performance of the various algorithms.

The four algorithms for this project are:

> **Opt** – Simulate what the optimal page replacement algorithm would choose if it had perfect knowledge
>
> **Clock** – Use the better implementation of the second-chance algorithm
>
> **FIFO** – Implement first-in, first-out
>
> **NRU** – Pick a not recently used page using the R and D bits.

You may write your program in C/C++, Java, Perl, or Python as long as it runs on thoth.cs.pitt.edu.

Implement a page table for a **32-bit** address space. All pages will be **4KB** in size. The number of frames will be a **parameter** to the execution of your program.

Project Details

You will write a program called `vmsim` that takes the following command line:

```
./vmsim –n <numframes> -a <opt|clock|fifo|nru> [-r <refresh>] <tracefile>
```

The program will then run through the memory references of the file and display the action taken for each address (hit, page fault – no eviction, page fault – evict clean, page fault – evict dirty).

When the trace is over, print out summary statistics in the following format:

```
Algorithm: Clock
Number of frames:       8
Total memory accesses:  1000000
Total page faults:      181856
Total writes to disk:   29401
```

Implementation

We are providing three sample memory traces. The traces are available at `/u/OSLab/original/` in the files `bzip.trace.gz, swim.trace.gz`, and `gcc.trace.gz`

Each trace is gzip compressed, so you will have to copy each trace to your directory under /u/OSLab/username and then decompress it like:

```
gunzip bzip.trace.gz
```

In the resulting trace file is a sequence of lines, each having a memory address in hexadecimal followed by a R or W character to indicate if that access was a read or a write. For example, gcc.trace starts like this:

```
0041f7a0 R
13f5e2c0 R
```

If you are writing in C, you may parse each line with the following code:

```
unsigned int address;
char mode;

fscanf(file, "%x %c", &addr, &mode);
```

Note

Implementing OPT in a naïve fashion will lead to unacceptable performance. It should not take more than **5 minutes** to run your program.

Write Up

For NRU, you have a refresh parameter to set. Try to find a good refresh period that works well. You do not need to find the absolute minimum, just approximately how long to wait. Plot your results in a graph and discuss why your choice of refresh seemed to be the best.

For each of your four algorithms (with NRU using the proper refresh you determined), describe in a document the resulting page fault statistics for 8, 16, 32, and 64 frames. Use this information to determine which algorithm you think might be most appropriate for use in an actual operating system. Use OPT as the baseline for your comparisons.

For FIFO, with the three traces and varying the total number of frames from 2 to 100, determine if there are any instances of Belady's anomaly. Discuss in your writeup.

File Backups

One of the major contributions the university provides for the AFS filesystem is nightly backups. However, the `/u/OSLab/` partition on thoth is not part of AFS space. Thus, any files you modify under your personal directory in `/u/OSLab/` are not backed up. If there is a catastrophic disk failure, all of your work will be irrecoverably lost. As such, it is my recommendation that you:

Backup all the files you change under `/u/OSLab` or QEMU to your `~/private/` directory frequently!

Loss of work not backed up is not grounds for an extension.

Requirements and Submission

You need to submit:

- Your well-commented program's source
- A document (.DOC or .PDF) detailing the results of your simulation as described above
- **DO NOT** submit the trace files!

Make a tar.gz file named `USERNAME-project3.tar.gz` and upload it to CourseWeb by the deadline.

Grading Sheet/Rubric

Item	Grade
vmsim program main skeleton	10%
OPT implementation	15%
CLOCK implementation	15%
FIFO implementation	15%
NRU implementation	15%
Writeup (Determining the optimal value of the refresh parameter)	10%
Writeup (The graph plotting the number of page faults versus the number of frames and your conclusions on which algorithm would be best to use in a real OS)	10%
Writeup (FIFO Analysis)	10%