Lab 5: Cache Lab – Understanding Cache Memories

Daeyeon Kim and Yejin Lee
Architecture & Code optimization (ARC) Lab

November 27th, 2018

Overview

In this lab,

- You can understand the impact that cache memories can have on the performance of your C programs

Part A: Writing a Cache Simulator

- With arbitrary size and associativity

Part B: Optimizing Matrix Transpose

- That causes as few cache misses as possible

Configuration

You need Linux environment

Install valgrind tool

- In martini server, valgrind is already installed

```
$> sudo apt-get install valgrind
```

Download Lab5.tar from eTL

```
$> tar -xvf Lab5.tar
$> cd Lab5
$> make clean; make
```

Part A: Writing a Cache Simulator

Your Task: To implement a cache simulator producing the same output of reference simulator (=`csim-ref`)

Reference cache simulator can simulate a cache with

- Arbitrary cache size
- Arbitrary associativity
- LRU replacement policy

File to modify (and submit): `csim.c`

Part A: Writing a Cache Simulator

At `traces/yi.trace`,

```
../Lab5$> cat traces/yi.trace
L 10, 1
M 20, 1
L 22, 1
S 18, 1
L 110, 1
L 210, 1
M 12, 1
../Lab5$>
```

Part A: Writing a Cache Simulator

To test your code,

```
../Lab5$> make clean; make
…
gcc -g -Wall -Werror …
../Lab5$> ./test-csim
                        Your simulator     Reference simulator
Points (s,E,b)    Hits  Misses  Evicts    Hits  Misses  Evicts
     0 (1,1,1)       0       0       0       9       8       6  traces/yi2.trace
     0 (4,2,4)       0       0       0       4       5       2  traces/yi.trace
     0 (2,1,4)       0       0       0       2       3       1  traces/dave.trace
     0 (2,1,3)       0       0       0     167      71      67  traces/trans.trace
     0 (2,2,3)       0       0       0     201      37      29  traces/trans.trace
     0 (2,4,3)       0       0       0     212      26      10  traces/trans.trace
     2 (5,1,5)       0       0       0     231       7       0  traces/trans.trace
     0 (5,1,5)       0       0       0  265189   21775   21743  traces/long.trace
     2

TEST_CSIM_RESULTS=2
../Lab5$>
```

Part B: Optimizing Matrix Transpose

Your Task: To optimize `matrix transpose` program

- On a 1KB direct mapped cache with a block size of 32 bytes

Files to modify (and submit): `trans.c`

- Write your code in "`transpose_submit()`" function

To test your code

```
$> make clean; make
$> ./test-trans -M m -N n
```

where *m* and *n* (≤ 256) are positive integers

Part B: Optimizing Matrix Transpose

To test your code,

```
../Lab4$> make clean; make
…
gcc -g -Wall -Werror …
../Lab4$> ./test-trans -M 16 -N 16

Function 0 (1 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 0 (Transpose submission): hits:15, misses:22, evictions:20

Summary for official submission (func 0): correctness=1 misses=22

TEST_TRANS_RESULTS=1:22
../Lab4$>
```

Grading Policy

Working correctly: 100 points

- Part A: 51 points
- Part B: 49 points
- For more details, see cachelab.pdf document in eTL
- You can check your score by this command below

```
../Lab5$> ./driver.py
```

Penalty for late submission: -20 % per every 24 hours

Submission Guideline

Make an archive file, "`Lab5.tar`"

containing "`csim.c`" and "`trans.c`"

```
../Lab5$> tar -cvf Lab5.tar csim.c trans.c
```

Upload on eTL with `Lab5.tar` as attachment.

Due date is Dec 13th (Thu) 23:59 PM (in three weeks!)