ECEC-412/621

Project 3: Implementing Cache Replacement Policy and evaluating its performance using AI Workloads

Instructor: Anup Das

1 Introduction

This project is intended to be a comprehensive introduction to cache replacement policy. There are two parts to this project:

- In part one, you will evaluate the performance of the Least-recently Used (LRU) replacement policy with three AI workloads from SPEC CPU 2017.
- In part two, you will implement a Least-frequently Used (LFU) replacement policy and evaluate its performance against part one.

2 Cache Replacement Policy Framework Overview

1. You will be working under directory *Computer-Architecture-Teaching/C621/Cache_Policy*. To navigate to the directory:

   ```
   $ cd Computer-Architecture-Teaching/C621/Cache_Policy/
   ```

2. To compile and run the simulator:

   ```
   $ make
   $ ./Main sample.mem_trace
   ```

3. You should be able to see the following output:

   ```
   Hit rate: 42.580000%
   ```

4. *sample.mem_trace* is part of the *leela* AI workload from SPEC CPU 2017. To take a look at the format of the trace file:

   ```
   $ tail -10 sample.mem_trace
   1 94009581925312 89824316667136 L
   1 94009581924100 89824316666944 L
   1 94009581932431 139713177052544 L
   2 94009581932431 139686110733696 L
   2 139693221530966 139686111208832 S
   0 94009582105510 132740395877696 L
   0 139693221530353 103162374522176 S
   1 94009582105510 139713177052480 L
   1 139693221530949 139713176905024 S
   2 94009582105510 139686110733632 L
   ```

   Each entry is composed of four components:

- Core ID: the core that loads/stores this cache block.
- Program Counter: the PC of the instruction that loads/stores this cache block.
- Cache Block Address: the physical address of the loaded/stored cache block.
- Load or Store.

3 Cache Replacement Policy Framework Configuration

The simulator models a set-associative last-level cache (LLC) with a LRU replacement policy. To configure the simulator:

1. Open *Cache.c*:

   ```
   $ vim Cache.c
   ```

2. You are encouraged to try different combinations of *cache_size* and *assoc*, and observe the changes of cache performance (hit rate). Please keep the cache block size to 64.

   ```
   const unsigned block_size = 64; // Size of a cache line (in Bytes)
   const unsigned cache_size = 128; // Size of a cache (in KB)
   const unsigned assoc = 4;
   ```

3. Re-compile and run the simulator:

   ```
   $ make clean
   $ make
   $ ./Main sample.mem_trace
   ```

4 Experiment

1. AI workloads (Memory Traces): `https://www.dropbox.com/sh/8fee5t192yche7s/AADr-T_7GKE185ye-CAAROOEa?dl=0`.
2. The simulator adopts LRU replacement policy by default. Simulate all the AI workloads with different combinations of *cache_size* and *assoc*. What combinations do you try and what do you find?
3. Implement a Least-frequently Used (LFU) replacement policy. Evaluate your policy against LRU. Which policy performs the best?

5 Submission

1. Summarize your experiment in Section 4. Compile your report in PDF format.
2. All the source codes.
3. Zip above and submit through Bblearn.