Computer Architecture

Project 3

Understanding gem5 Cache structure

Source: http://www.gem5.org/Replacement_policy

You can find the different functions in cache replacement policies from the website above. There are five main function called in each cache replacement policy.

A replacement policy consists of a reset(), touch(), invalidate() and getVictim() methods. Each of which handles the replacement data differently.

1. reset() is used to initialize a replacement data (i.e., validate). It should be called only on entry insertion, and must not be called again until invalidation. The first touch to an entry must always be a reset().
2. touch() is used on accesses to the replacement data, and as such should be called on entry accesses. It updates the replacement data.
3. invalidate() is called whenever an entry is invalidated. It makes the entry as likely to be evicted as possible on the next victim search.
4. getVictim() is called when there is a miss, and an eviction must be done. It searches among all replacement candidates for an entry with the worst replacement data.

Note: Please read the above functions carefully to understand how each policy is implemented.

Understanding LRU Policy

In the design of set associative caches, an important design parameter to analyze - is the block replacement policy. Block replacement policy essentially determines which block (out of all the blocks in the set) must be evicted from the cache, when it is time to bring in new data.

One of the most popular cache replacement policies is Least Recently Used (LRU). In this scheme, we evict the block that has been unused for the longest period. While it seems to be one of the most intuitively promising block replacement schemes, implementation complexity is its main disadvantage. LRU must maintain an access history for each block, which slows down the cache. Thus, most caches implement only an approximation of LRU.

Note: You will find the cache replacement policies implementation in the gem5 source directory – <path to gem5-cse-ca>/src/mem/cache/replacement_policies/lru.cc and lru.hh

Problem 1 [7 points] – Understanding and Implementing LRU-IPV

Daniel A. Jiménez proposed the Insertion and Promotion for Tree-Based PseudoLRU Last Level caches in 2013. Link to paper http://taco.cse.tamu.edu/pdfs/p284-jimenez.pdf. You need to understand the Figure 3 in the paper and implement the graph with IPV [0 0 1 0 3 0 1 2 1 0 5 1 0 0 1 11 13], same as the paper.

You have to create two files namely lru_ipv.cc and lru_ipv.hh in the – <path to gem5>/src/mem/cache/replacement_policies/ directories. Have the five main functions namely reset(), touch(), invalidate() and getVictim() methods to implement the IPV. You can hard code the IPV [0 0 1 0 3 0 1 2 1 0 5 1 0 0 1 11 13] in either the hh or cc file.

Change 1: Change to ReplacementPolicies.py

Name your class as `LRUIPVRP()` and add the following in your **`src/mem/cache/replacement_policies/ReplacementPolicies.py`**

```
class LRUIPVRP(BaseReplacementPolicy):

    type = 'LRUIPVRP'

    cxx_class = 'LRUIPVRP'

    cxx_header  = "mem/cache/replacement_policy/lru_ipv.hh"
```

Change 2: Change to SConscript

Add the following line at the end of src/mem/cache/replacement_polices/SConscript

```
Source('lru_ipv.cc')
```

Note: After all the above modifications recompile gem5 by running the following command:

```
$> scons build/ALPHA/gem5.opt
```

Problem 2 [3 points] – Running your implementation

The gem5 provided to you has been modified to include the command line to call the replacement policies. Command line option is --repl_policy. You have to run the LRU IPV implemented gem5 for dijkstra and basicmath. Here is the table for execution

Config	L1 cache	L2 cache	L2 associativity	Benchmarks
1	32 Kb data cache, 32 Kb Instruction cache	256 Kb	16	• dijkstra • basicmath
2	32 Kb data cache, 32 Kb Instruction cache	1 Mb	16	• dijkstra • basicmath

The gem5 commandline options you will be using for the assignment is the following

```
build/ALPHA/gem5.opt --outdir=Problem2/basicmath-Conf1
configs/example/se.py --cpu-type=AtomicSimpleCPU --caches --
l1i_size=32kB --l1d_size=32kB --l2cache --l2_assoc=16 --
repl_policy="LRUIPVRP()" --l2_size=256kB -c
<path_to_benchmarks>/basicmath/basicmath_small
```

Project Deliverables

1. For this project, you will create a patch file, which is portable and easy to apply your gem5 modifications and remove them by the graders. You will have to follow the following instructions to create a correct patch.
    a. If you have followed the getting started with gem5 document then you should have two copies of gem5. One with name gem5 and one with gem5-cse-ca. You should have modified the gem5-cse-ca directory.
    b. To create a correct patch run the following command;

```
diff -ruN gem5/src/mem/cache/replacement_policies/ <path_gem5-
cse-ca>/src/mem/cache/replacement_policies/ > project3.patch
```

Note: Make sure the "N" in -ruN is uppercase. You can open project3.patch and locate all your modifications to ReplacementPolicies.py, SConscript, lru_ipv.hh and lru_ipv.cc with a "+" sign at the beginning of each line modified.

2. Like project 1, you will submit the config.ini, config.json and stats.txt along with the patch file created. The file directory should look like this

```
\---Project3-Submission
```

```
+---Problem1
|   +---project3.patch
|
\---Problem2
    +---dijkstra-Conf1
    |       config.ini
    |       config.json
    |       stats.txt
    |
    +---dijkstra-Conf2
    |       config.ini
    |       config.json
    |       stats.txt
    |
    +---basicmath-Conf1
    |       config.ini
    |       config.json
    |       stats.txt
    |
    \---basicmath-Conf2
            config.ini
            config.json
            stats.txt
```

Submission Instructions

1) Submit single zip file containing all the files. Please follow the naming conventions correctly.

2) In case where you are doing project in a group, only one group member should make submission through blackboard. All the team members of the group will receive the same score.

Rubric

1. **Problem 1 [7 points]**
    a. Correct patch file, grader should have no problem in applying the patch and running gem5 with your implementation [2 points]
    b. Correct implementation of LRU IPV according to the paper, with comments in the cc and hh files. [3 points]
    c. Correct class names, according to the Project deliverables [2 points]
2. **Problem 2 [3 points]**
    a. Correct configuration from the table from config.ini file. [1 points]
    b. Correct stats in the stats.txt file [2 points]