CS622A

Advanced Computer Architecture

Assignment 1

Multi Level Cache Simulator

GROUP 16

Aditya Rohan
160053

Aniket Pandey
160113

Instructor:
Dr. Mainak Chaudhury

August 31, 2019

1 Introduction

A program to simulate L1 cache misses through L2, L3 cache hierarchy for various applications. The simulator is programmed for 3 different inclusion policies, i.e *Inclusive*, *Exclusive* & *Non-Inclusive Non-Exclusive* (*NINE*).

- **Part 1:** 8-Way L2 cache (LRU), 16-Way L3 cache (LRU)
- **Part 2(a):** 8-Way L2 cache (LRU), Fully-Associative L3 cache (LRU)
- **Part 2(b):** 8-Way L2 cache (LRU), Fully-Associative L3 cache (MIN)

2 Instructions to Run the Simulator

The commands to compile and run the code are specified in a `Makefile`. The details are explained below.

2.0.1 PART 1

1. `make process`: Reads the traces and prepares an input format for simulator. **Note:** The miss traces MUST be present in a directory named *traces* in the root folder.
2. `make simulate`: Compiles and runs the cache simulator for **part 1**. L2, L3 cache hits and misses are printed in stdout.

Note: `make part1` runs the above two commands together.

2.0.2 PART 2

1. `make fa-lru`: Run the simulator for Fully associative L3 cache with LRU cache replacement policy.
2. `make fa-min`: Run the simulator for Fully associative L3 cache with Belady's MIN cache replacement policy.

Note: `make part2` runs the above two commands together.

3 Simulation Results

3.1 PART 1: Set Associative, LRU eviction policy

INCLUSIVE	**L2**		**L3**	
	Hits	Misses	Hits	Misses
bzip2	5259461	5398166	3951778	1446388
gcc	11574350	3036461	1663059	1373402
gromacs	3094660	336851	166320	170531
h264ref	1378895	969678	627532	342146
hmmer	1766344	1743421	1352195	391226
sphinx3	1933098	8820349	612987	8207362

EXCLUSIVE	**L2**		**L3**	
	Hits	Misses	Hits	Misses
bzip2	5260051	5397576	4508355	889221
gcc	11581002	3029809	1786985	1242824
gromacs	3094787	336724	177422	159302
h264ref	1382949	965624	821943	143681
hmmer	1774443	1735322	1435276	300046
sphinx3	1938317	8815130	1594354	7220776

NINE	**L2**		**L3**	
	Hits	Misses	Hits	Misses
bzip2	5260051	5397576	3951730	1445846
gcc	11581002	3029809	1663561	1366248
gromacs	3094787	336724	166265	170459
h264ref	1382949	965624	632041	333583
hmmer	1774443	1735322	1358978	376344
sphinx3	1938317	8815130	609986	8205144

3.2 PART 2: Fully Associative L3 Cache

INCLUSIVE	LRU			Belady MIN		
L3	**Cold**	**Capacity**	**Conflict**	**Cold**	**Capacity**	**Conflict**
bzip2	119753	1241648	84987	119753	417083	909552
gcc	773053	596871	3478	773053	166236	434113
gromacs	107962	61406	1163	107962	35292	27277
h264ref	63703	272177	6266	63703	47902	230541
hmmer	75884	301140	14202	75884	77563	237779
sphinx3	122069	8265179	-179886	122069	2946511	5138782

4 Simulation Analysis

Following are some of the observations on the simulation results.

4.1 Part 1

4.1.1 L2 Hits/Misses (Exclusive) == L2 Hits/Misses (NINE)

This is observed because at any point in time, the addresses that are present in case of Exclusive policy are same as that of NINE, albeit in different configuration. We can analyze all possibilities to prove this:

- **L2 Hit:** Nothing changes here.
- **L3 Hit:** The block gets added to L2 Cache. Although the block is invalidated from L3 in exclusive case.
- **L3 Miss:** The block is brought from memory and added to both L2 and L3 cache.

Hence, in all the cases, if the configuration is same after *kth* steps, it will be same after *(k+1)th* step.

4.1.2 L3 Misses: Inclusive > ($\approx$) NINE >> Exclusive

At any given point, L3 cache in case of Exclusive policy will have most absolute space (due to non-similarity of data in L2 and L3). This explains why L3 misses in Exclusive are significantly lesser.

There isn't much difference in L3 misses for Inclusive and NINE cases, although Inclusive is slighly more. This can be attributed to the fact that eviction from L3 also evicts from L2 for Inclusive policy, which is not the case with NINE. So, an access to a block evicted from L3 would encounter a hit in L2 for NINE but would unnecessarily miss in both L2, L3 in Inclusive case.

4.1.3 L2 Misses: Inclusive > Exclusive == NINE

This is a serious problem with Inclusive policy, as mentioned in problem statement. If a block recieves constant hits in L2 cache, its rank in the LRU indexing will be quite high. As a result, the block will normally never be evicted from L2. But its age in L3 will drop significantly and might even become victim to an eviction. Following the Inclusive policy, it will also have to be removed from L2 which would unnecessarily increase L2 misses.

4.2 Part 2

4.2.1 Method of categorizing FA L3 misses

Cold: Number of unique block references.
Capacity: FA-L3 misses - Cold misses.
Conflict: 16-way L3 misses - FA-L3 misses.

1. Number of capacity misses for LRU eviction policy are more than that of Belady's policy since Belady is the most optimal solution for block replacement.

2. Number of conflict misses for *sphinx3* trace are negative (-179886). For some uncommon cases, the approach of calculating conflict misses from cold and capacity of fully-associative cache can result in a negative number [1].

References

[1] Yan Solihin. *Fundamentals of Parallel Multicore Architecture*. Chapman and Hall/CRC, 2015.