Cache Simulator Report

Boyi Zhang, Zeyang Yu

March 2024

1 Set-associative LRU (least-recently-used) replacement cache simulator

1.1 Implementation

This cache simulation uses a combination of doubly-linked lists and arrays to model a set-associative cache with an LRU replacement policy. The global cache contains an array, each element representing a double-linked lists. In the doubly-linked lists, head node represent the most recently used memory; the tail node represent the least recently used memory. Every time cache try to access an address, program will calculate the tag and index from address then find the corresponding doubly-linked list. Then traverse the linked-list to check if memory exists in the sets. There are three cases in simulation:

- tags in the linked lists, meaning a cache hit, moving the node to the head
- tags does not exists in the linked lists ,and the number of nodes is less than the size of the cache sets. A new node will be created, the new node will become the head of the doubly linked lists.
- tags does not exists in the linked lists, and number of nodes is equal to the size of the cache sets. Then the tail of the doubly-linked list is removed. A new node is created and become

the head of the doubly linked lists.

The primary control flow is through the simulation's start (‘sim_start‘), access (‘sim_access‘), and finish (‘sim_finish‘) functions, which manage the setup, operation, and teardown of the simulation, respectively.

1.2 Test result

There are two test units: cyclic and sawtooth. Each test data size is 10^6 4-byte with 10 repeats. Table1 shows the test results. In LRU, the miss count of sawtooth access is much lower than the cyclic access.

	Cyclic	Sawtooth
Miss Count	1,250,000	938,704
Run Time	0m0.287s	0m0.283s

Table 1: Comparison of Cache Misses and Run Time in LRU

2 Team Work

We build two types of cache simulator: Set-associative LRU replacement cache and Set-associative MRU (most recently used) replacement cache. My team's reports focuses on the question: with MRU replacement policy, how size of cache line will affect locality. On the other hand, this report will focus on how set-associativity will affect locality with the LRU replacement policy.

3 Comparison: Set-Associativity

The reports focuses on how set associativity affects locality (measured in miss counts) in the Set-associative LRU replacement cache.

3.1 experiment

In the experiments, simulator is the same as the one using in part one; the data size is fixed to 10^4 4-byte ints with 10 repeats; there are 8 bytes is each cache lines; 8 sets in the whole cache. The testing includes set associativity in $2, 4, 16, 256$. The simulator is tested on both cyclic and sawtooth data access.

Associativity	Cyclic Miss Count	Sawtooth Miss Count
2	10,000	9,696
4	10,000	9,392
8	10,000	8,784
16	10,000	7,568
32	10,000	5,136
64	500	500

Table 2: Comparison of Cyclic and Sawtooth Miss Counts for Different Associativity

3.2 Conclusion

For lower levels of associativity (2, 4, and 16 and 32), the cyclic miss count remains constant at 10,000 misses. This suggests that for this particular access pattern, increasing the associativity within this range does not improve the cache's ability to reduce misses.

The sawtooth access pattern shows a gradual decrease in miss counts as associativity increases from 2 to 32, indicating that this access pattern benefits from higher associativity

For both cyclic and sawtooth access, a dramatic improvement is observed when the associativity increases to 64, with the miss count dropping to 500. This indicates a threshold beyond which increased associativity substantially enhances the overall locality. Because in 64 set associativity the cache size is larger than the data sets, meaning that all the data is loaded in the cache, threshold effects occurs. However, this is not realistic in improving locality in the actual practice. Because, in real work scenario, the data access will be much larger than the cache size. It is impossible to

make cache as large as the data size.