Richard Wu | wuricha8 | 1008078296
Christian Vedtofte | vedtofte | 1012872125

ECE552 Lab 4 Report

Question 1: Next-line Prefetcher Benchmark

Our microbenchmark (mbq1.c) tests the next-line prefetcher with two scenarios. In Scenario 1, the prefetcher will always fetch useless data, so cache accesses will always miss; in Scenario 2, the prefetcher will always fetch useful data, so cache access will always hit.

- Scenario 1: We iterate and access data two blocks away each time. Since the next-line prefetcher only fetches the next block, we will never fetch the right data.
- Scenario 2: We iterate and access data one block away each time. So the next-line prefetcher will fetch the right data.

Scenario 1 runs 100,000 times and Scenario 2 runs 200,000 times. This means about 100,000 cache misses and 200,000 hits; so an expected data cache L1 miss rate of about 33%. Simulation gives us 203768 hits and 100022 misses and a miss rate of 32.92%. These numbers are very close to our expectations.

Question 2: Stride Prefetcher Benchmark

Our microbenchmark (mbq2.c) tests the stride prefetcher with two scenarios:

- Scenario 1: We iterate and access data but alternate between incrementing the memory address by one and two block sizes. Each access is performed by the same instruction, so one RPT entry is used. In this way, a consistent stride can't be found, so the stride prefetcher will have this entry be in the 'No Prediction' state forever. Thus no useful data will be fetched.
- Scenario 2: We now have two load instructions. The first load increments its memory to access by two block sizes and the second load increments its memory to access by one block size. Thus, we will have two separate RPT entries and consistent strides will be found for each. So the stride prefetcher will always fetch useful data for these two loads.

Scenario 1 runs 133 thousand times with expected 133 thousand cache misses. Scenario 2 runs 200 thousand times with expected 400 thousand cache hits. This means an expected 25% cache miss rate. Simulation gives us 394394 hits and 142729 misses and a miss rate of 26.6%. These numbers are very close to our expectations.

Question 6: Open Ended Prefetcher Explanation and Microbenchmark

Open Ended Prefetcher Design

Our open ended prefetcher is a global history buffer (GHB) based design. Our structures include a 5-entry GHB, which is implemented as a FIFO queue and a "delta" table that stores information about recently observed address differences (deltas). On every call to this prefetcher we do the following:

1. Calculate the (new) delta between the current access address and the previous one. Update the delta table with this information. Evict the tail of GHB and insert a new entry at the head, setting its delta to what was calculated.

2. Search the GHB in reverse for an entry that matches the deltas in the delta table. If an entry's delta matches the previous delta (call this entry A) and the entry before A (which we will call B) has a delta matching the new delta, pick the entry before B (let's call it C).
3. Use the delta of entry C as the stride to conduct a prefetch on.

Essentially, the algorithm searches if there is a sequence of two accesses in its history buffer with strides that match, then uses the stride from right before that to get the prefetch address.

Open Ended Prefetcher Microbenchmark

Our microbenchmark (mbq6.c) tests our open-ended prefetcher with two scenarios:

- Scenario 1: We iterate and access data, incrementing by 1, 1, 2 block sizes repeatedly. In this way, the global history buffer (GHB) is able to detect a sequence of two consecutive matching strides and can accurately predict the next one. Thus, the prefetcher will always fetch useful data.
- Scenario 2: We iterate and have two load instructions that increment the address they access by 1 and 2 block sizes, respectively. In this way, the GHB is not able to find a pattern and so it will not be able to prefetch. Thus we expect to always get cache misses.

Scenario 1 runs 150 thousand times with expected 150 thousand cache hits. Scenario 2 runs 200 thousand times with 400 thousand cache accesses (from two load instructions per iteration) and misses. This means an expected 72.72% data cache miss rate. Simulation gives us 153854 hits and 399936 misses and a miss rate of 72.22%. These numbers are very close to our expectations.

Open Ended Prefetcher Area Overhead and Access Time Assessment

Our open-ended prefetcher consists of a GHB and a delta table. The GHB is simply a structure with 5 entries, each entry holding a 'delta' which is just a number. The delta table is just a structure with 2 entries, tracking the current delta and the previous delta. All in all, the entire prefetcher can be structured as a RAM-style structure with 7 entries for numbers, where each entry is 4 bytes (32 bit words), for a total size of 28 bytes. Using the pureRAM template for CACTI from Lab 2 (with a minimum size of 64 bytes), we get the following results:

- Access time(ns): 0.118515
- Area(mm^2): 0.00025572

To compare, we made a CACTI config (using the cache template) for the cache consistent with q6.cfg:

- DL1 cache: 64 sets * 64 blk size * 4 assoc = 16384 byte size
- 64 blk size → 64*8 = 512 bit bus width
- 64 blk size → 6 offset bits
- 64 sets → 6 index bits
- Tag bits = 32 - 6 - 6 = 20 bits

The results for this cache are:

- Access time(ns): 0.447214

Richard Wu | wuricha8 | 1008078296
Christian Vedtofte | vedtofte | 1012872125

- Area(mm^2): 0.055099

Thus the area overhead is 0.46% which is negligible. However, the access time overhead is 26.5%. Further, since the GHB may need to be accessed multiple times per prefetch, this adds more overhead. So this prefetcher adds considerable latency to this cache.

Question 3

Average Memory Access Time for Compress Benchmark

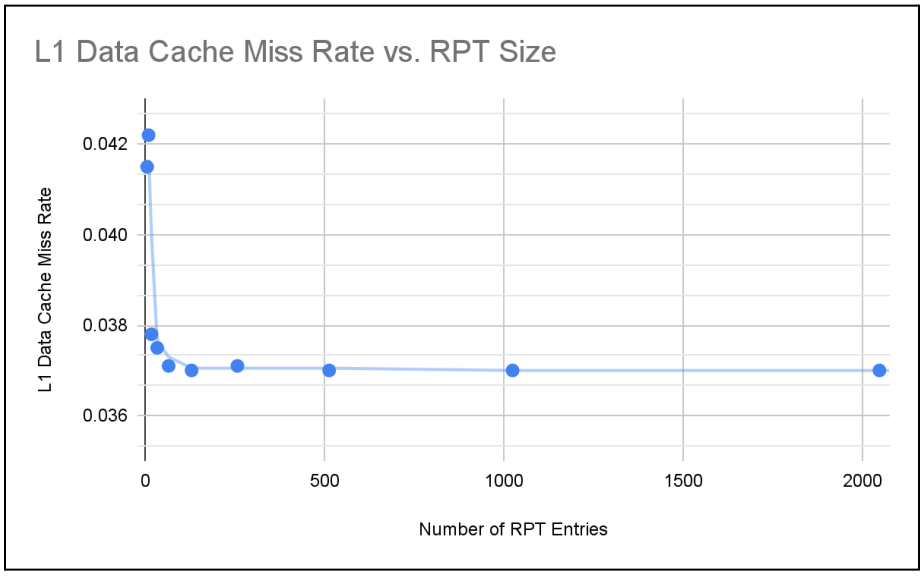
Config	L1 Miss Rate	L2 Miss Rate	Average access time
No Config	0.0416	0.1140	1.801216
Next-Line	0.0419	0.0838	1.6931098
Stride	0.0378	0.0516	1.5157432

We have that L1 access takes 1 unit of time, L2 access takes 10 units of time, and memory hit takes 100 units of time. So the average access time is:

Average Access Time = 1 * (1-(L1 miss rate)) + 10 * (L1 miss rate)*(1-L2 miss rate) + 100 * (L1 miss rate)*(L2 miss rate)

Question 4

We varied the RPT size for the Stride Prefetcher from 4 to 2048 entries, doubling the size each time. The L1 data cache miss rate is used as the metric for performance since L1 data cache access has a direct impact on memory access speed. A lower miss rate means a better performance. Below is a plot of L1 data cache miss rate vs RPT size for the Compress benchmark.



Richard Wu | wuricha8 | 1008078296
Christian Vedtofte | vedtofte | 1012872125

From the trend, we see that performance gains are achieved for up to around 64 RPT entries, then the miss rate converges to some steady state value (which we found to be 0.037). So larger RPT sizes don't provide significant performance gains. This makes sense because there is some minimum RPT size where all instructions that index into it will be accommodated, larger tables will only result in unused space.

Question 5

Two useful, related, statistics would be the number of useful and not-useful prefetches. Knowing these numbers would help us evaluate the performance of our prefetchers because we can know how often the cache is being polluted by useless prefetches vs how often prefetched data in the cache is being used. A higher number of useful prefetches is a good indicator of good performance.

Statement of Work

Both Richard and Christian worked on all three prefetchers. Christian worked more on testing the stride prefetcher and its microbenchmark. Christian also worked on researching algorithms to implement for our open-ended prefetcher. Richard worked more on writing the microbenchmarks for the next-line and open-ended prefetchers.