Laboratory 11/Problem Set 9
Memory Hierarchy: Caches

Computer Science 240

This lab explores the implications of caches in memory system performance, starting with experiments that evaluate real programs on real hardware and then moving to simulations.

Caches are managed solely by hardware. They are hidden from programmers by the ISA's abstraction that memory is simply a large array of bytes. While caches never change the meaning of a program (the result it computes), their impact on program performance can be observable and, in some cases, extreme.

This lab will:

1. Evaluate our earlier claim that two versions of a program that compute the same result and differ only in the order of two adjacent lines of code can have running time that differs by an order of magnitude.

2. Demonstrate that, while caches are semantically invisible to programs, their performance impact can be used to measure their dimensions.

Download and unzip ***cs240-cache-sleuth.zip*** from the Google group. This folder contains a number of Java files.

Use Dr. Java **on the lab machines** (do not use your own labtop) to open,compile, and run the programs. Also, close all other programs while you are running your experiments (because other processes also use the cache, and can interfere with your results).

Part 1: Cache Experiments on Real Hardware

Exercise 1: Examine methods ***experimentA*** and ***experimentB*** in ***CacheExperiment.java*** to make a hypothesis about performance.

1. Describe what ***experimentA*** and ***experimentB*** compute.
They both compute the (sum of array elements) * reps

2. How do their implementations differ?
The loops are switched. In A, the individual array element is accessed multiple times before moving on to the next repetition of the outer loop.

In B, the array elements are stepped through completely for each repetition of the outer loop.

3. On a machine that uses caches with a least-recently-used (LRU) replacement policy in the memory hierarchy, which of A or B is likely to run faster in some cases? What principle is involved? A will run faster becauseA has better spatial and temporal locality than B.

4. Make a hypothesis: How do the following factors affect the (relative) performance of A and B?

- The total cache capacity: C bytes

The array size: S elements If the cache is smaller than the array size, B will perform much more poorly than A.

- The number of repetitions: R The number of reps will multiply the delay experienced by B compared to A.

Exercise 2: Partially evaluate your hypothesis via timing experiments to estimate the capacity of the cache in the machine you are using. In all experiments below, repeat each individual experiment 5 times and record all results to account at least partially for variability in timing.

You will compile ***CacheExperiment*** once and run it several times for this step.

First, experiment with effects of array size.

1. Use a fixed number of repetitions and vary the size of the array in use.

    a. Run ***CacheExperiment*** with size 500,000 elements and 1,000 repetitions.

    b. Repeat the experiment above, increasing the array size by 100,000 each time until you reach 2,000,000.

2. On paper, plot the average time of A and B for each array size as a line chart.

3. One curve should exhibit a "knee," a point in a curve where the slope changes noticeably compared to the slope on either side. Which curve contains a knee?
B contains a 'knee', while A has a constant slope.

4. Do more experiments, varying size to find the most precise location of the knee that you can. Repeating your experiments to account for variability is important.

You may find that you eventually reach a point where results become too variable to continue.

5. As you “zoom in,” is the knee angular? noisy? rounded?   The knee seems rounded, and is not that definite.

Exercise 2: Now, experiment with effects of repetitions.

1. Use a fixed array size and vary the number of repetitions.

2. Plot the average time of A and B for each array size as a line chart.

3. Measurements will get noisy as you approach 1 repetition.

- Run ***CacheExperiment*** with size 10,000,000 elements and 1,000 repetitions.

- Repeat the experiment above with 500, 100, 50, and 10 repetitions.

- Answer the following:

a. Is there a knee in either curve? Yes, again in B.

b. How close are the running times of A and B at 10 repetitions? Very close.

c. What do you expect of the relative performance of A and B at 1 repetition? Why? They should be the same, because each elemnt of the array is accessed exactly once in both programs.

d. What are some reasons that the measures could get noisy approaching 1 repetition? The cache is shared with instructions, data, and other processes, so at one repetition it is unlikely that the time will be the same.

e. Finally, estimate the cache size of the machine. Use your hypothesis and the data you collected in these two series of experiments. Assume the cache uses a true LRU replacement policy and memory is byte-addressable. The cache size (in bytes) might not be a power of 2, but it is no more complicated than the sum of two powers of 2. It seems that the bend occurs around an array size of 1.5 million, and each element of the array is 4 bytes. So, that is 6MB, which is not a power of two, but which is the sum of 4MB and 2BM.