Department of Electrical & Computer Engineering
University of California, Davis

EEC270 – Computer Architecture
Winter 2018

Homework 2 - Part B

Due: February 9, 2018 (Maximum Score = 100)

Cache Optimization

Understanding the relationship between software and cache behavior is critical for writing high performance algorithms. Even a small increase in the L3 cache miss rate can significantly slowdown program execution. In this assignment, you will investigate how various implementations of a simple matrix multiplication interact with a computer's cache and affect execution time. A template C++ file is included with this assignment for your convenience. Refer to that for compilation instructions.

Valgrind

The open source software **Valgrind**[1] is a suite of profiling tools offering services from memory and thread error detection, to a cache profiler. This last tool, called **Cachegrind** is what we will use to profile the cache hit and miss rate of the matrix multiplication routine.

Valgrind is installed on the ECE department's Linux machines. While you can install Valgrind on your own system, you may find it easier to use the existing installation on these machines. You can either use those machines directly, or SSH into those machines. Instructions can be found at `http://www.ece.ucdavis.edu/support/remote/`.

For quick reference, if a compiled executable is titled `a.out`, Cachegrind can be run using the following terminal command:

```
valgrind --tool=cachegrind ./a.out
```

If `a.out` takes commandline arguments these are given at the end, just as if it was being run directly. More information can be found by executing

```
valgrind --help
```

[1]http://valgrind.org

Assignment

1. Either using the C++ template provided, or using your own C/C++ program, write functions to perform:

   - The naive matrix multiplication algorithm using three nested `for` loops.
   - A block matrix multiplication routine using at least five nested `for` loops.

   In a single graph, plot the execution time to perform the naive algorithm and blocked algorithm with block sizes

   $$2, 4, 8, 16, 32, 64 \tag{1}$$

   for the following size matrices:

   $$\begin{gathered} 128 \times 128 \\ 256 \times 258 \\ 512 \times 512 \\ 768 \times 768 \\ 1024 \times 1024 \end{gathered}$$

   What is the optimal blocking factor?

2. Use **Cachegrind** to track the L1 and L3 (reported as LL by Valgrind) data cache miss rate. Report the total miss rate, read miss rate, and write miss rate for the naive matrix multiplication algorithm and blocked algorithm with the block sizes in (1) for the 512×512 and 1024×1024 matrix sizes.

   What do you observe about the cache miss rates and the execution times? Why do you think the miss rates appear as they do?

3. Cachegrind allows you to change the associativity and sizes of the various caches. Repeat part 2 for a direct mapped L1 data cache and then for an L1 data cache that is twice the size of the cache on the machine you are using. What do you observe?

   Note: refer to `http://valgrind.org/docs/manual/cg-manual.html` for information on how to change the sizes and associativity of the simulated cache for cachegrind.