Michigan State University

CMSE/CSE 822 – Parallel Computing **Fall 2019**

Homework 2

Alexander Harnisch

---

1) Performance Modeling

a)

There is some missing information here and assumptions have to be made. However, it's not important since it only effects the numbers to plug in and get out, not the logic behind it.

The kernel performs 6 floating point operations (FLOP) in each iteration. Assuming no caching we have seven floating point number reading and one writing operation for each loop iterations. However, we can assume that the same variables are kept in the registers in each loop iterations, or at least in fast memory. Which means the entire y and z arrays both only have to be loaded once. Assuming single precision with 4 byte per float that translates to 12 byte of memory access and an arithmetic intensity I of

$$I = \frac{6\,\text{FLOP}}{12\,\text{byte}} = \frac{1}{2}\,\frac{\text{FLOP}}{\text{byte}}\,. \tag{1}$$

b)

In a simple roofline model for some I the critical peak performance π_{crit} is given by βI, where β is the peak memory bandwidth. So for $I = 0.5\,\frac{\text{FLOP}}{\text{byte}}$ from a) we get:

$$\pi_{\text{crit}} = 30\,\frac{\text{GB}}{\text{s}} \cdot \frac{1}{2}\,\frac{\text{FLOP}}{\text{byte}} = 15\,\frac{\text{GFLOP}}{\text{s}}\,. \tag{2}$$

So in case the processor's peak performance is greater than 15 GFLOP/s the kernel is compute bound, otherwise memory bound.

c)

A simple roofline model plot is given by Figure 1. The performance for an arithmetic intensity of $I = 0.5\,\mathrm{FLOP/byte}$ is $15\,\mathrm{GFLOP/s}$.

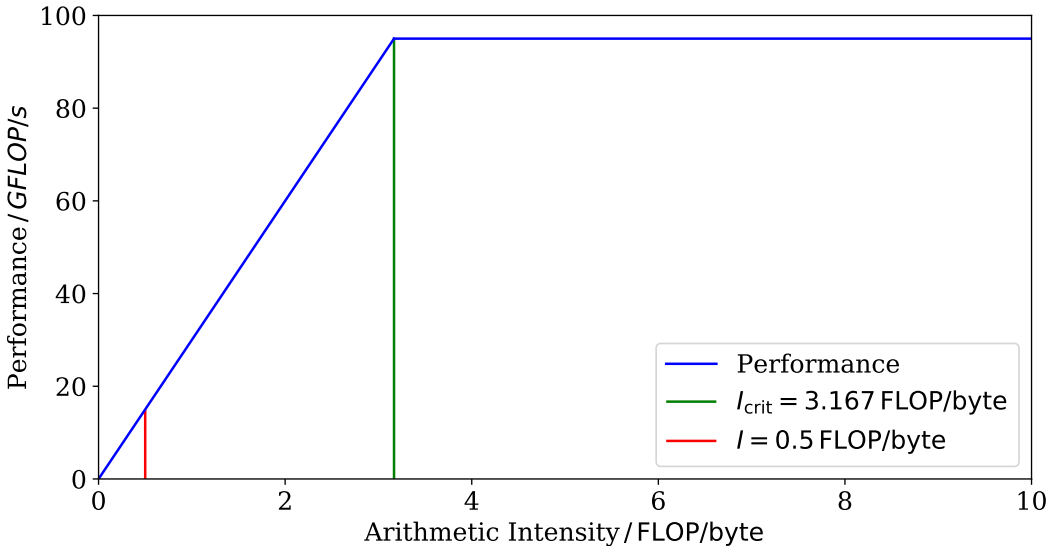

Figure 1: Simple roofline model.

2) Cache optimization: Matrix Vector Multiplication

a) Implementation

See code in repository.

b) and c) Performance Analysis and Cache Performance Measurement

For all of the testing in this subsection, scripts for fully automated job submission and data collection have been written in Bash and Python. Using this system, 1968 different sets of parameters (N, M and B) were tested. For each run the relative speedup as well as cache misses for L1, L2 and L3 were collected as variables of interest. All of the results are summarized in the following plots. The raw data can be found in the file *data.txt* in my repository, outside of the submission directory.

Figure 2 shows the speedup for all tested matrices in dependence of the blocking factor B. Each data point is the average speedup for all matrices with the corresponding B value.

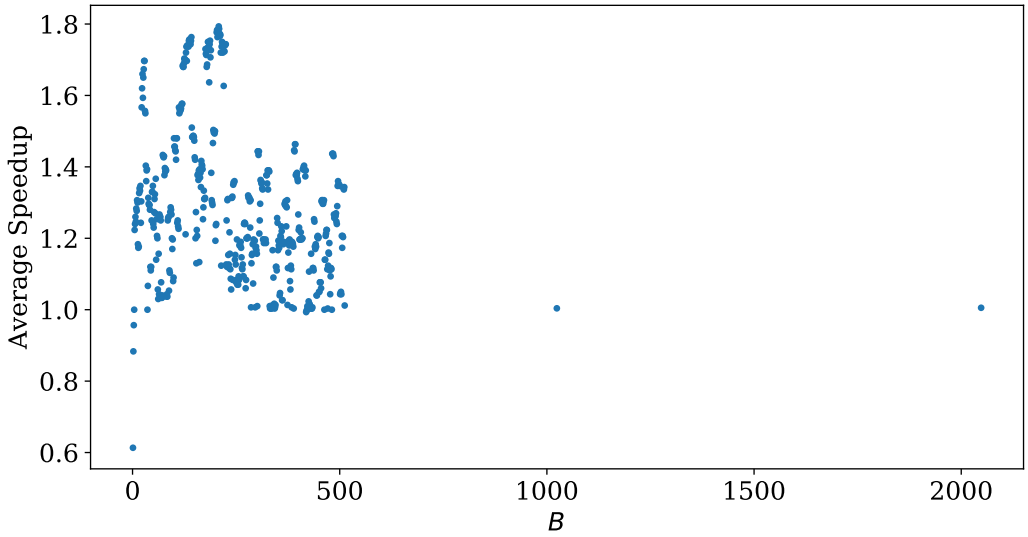

Figure 2: Average speedup for all tested matrices in dependence of the blocking factor B.

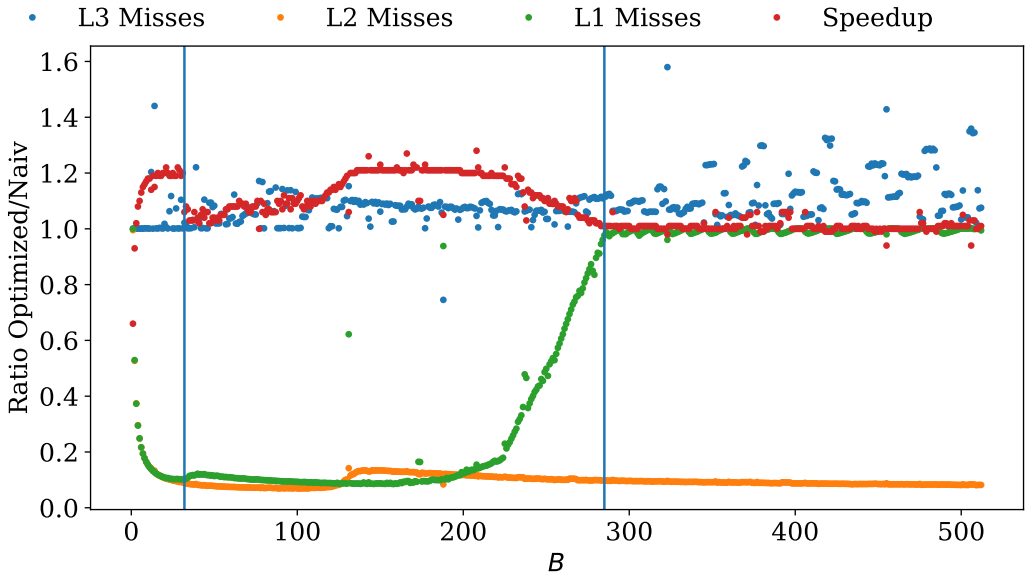

Figure 3: Results for a fixed matrix size of $N = M = 10000$ for different blocking factors B. For reference, the plot includes two blue vertical lines at $B = 32$ and $B = 285$.

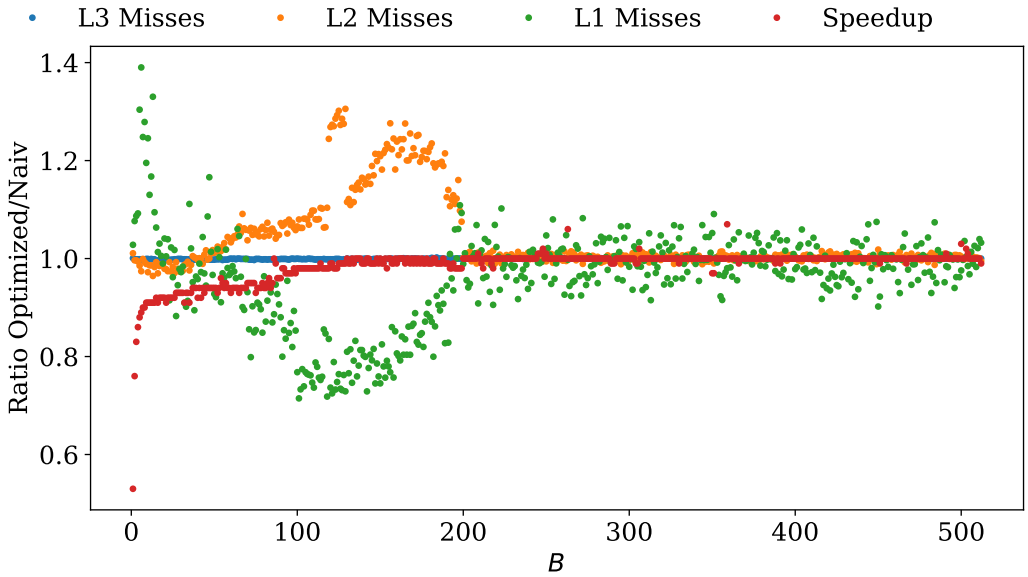

Figure 4: Results for a narrow matrix with $N = 100000$ and $M = 200$ for different blocking factors B.

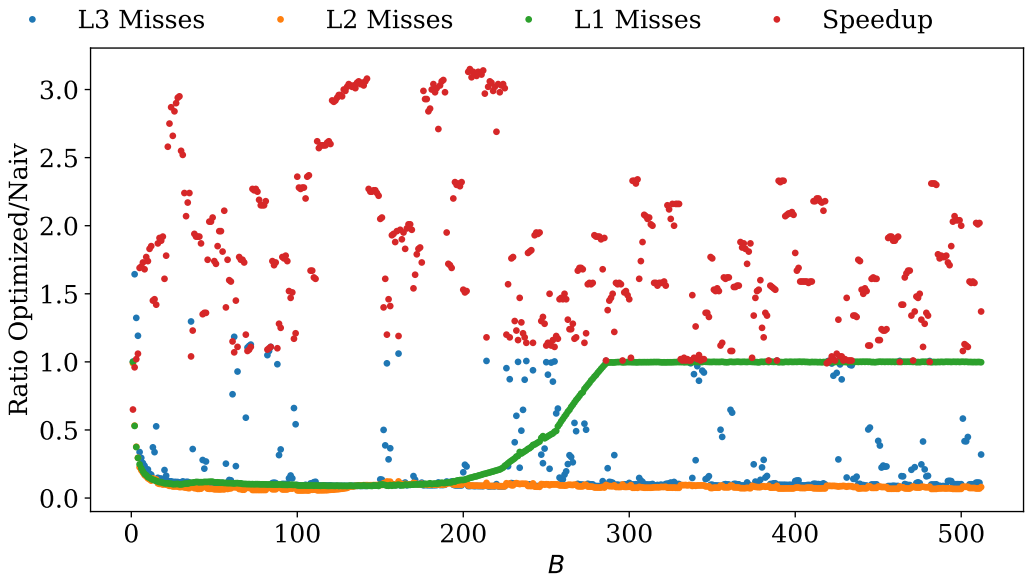

Figure 5: Results for a wide matrix with $M = 100000$ and $N = 200$ for different blocking factors B.

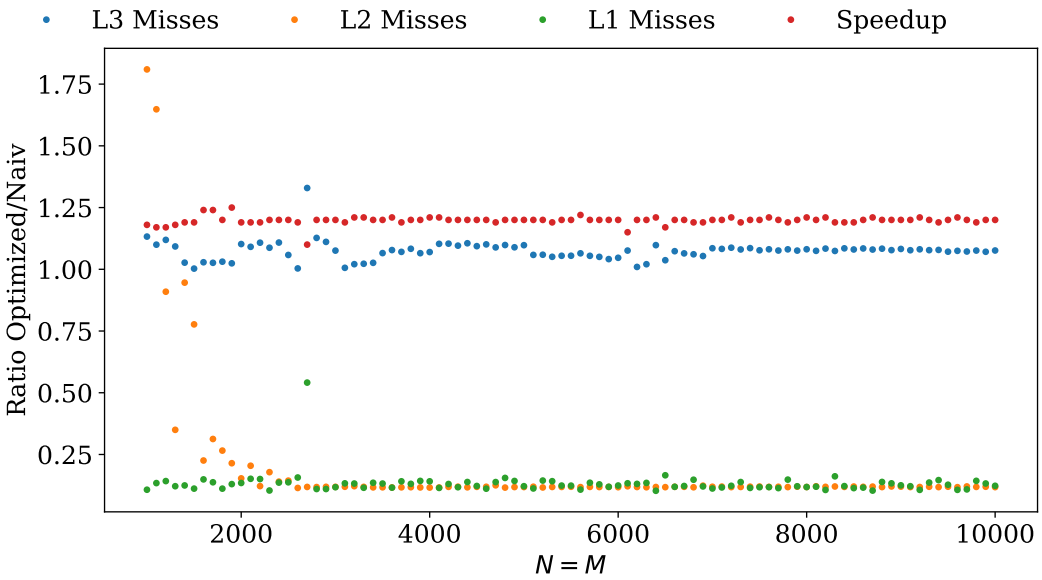

Figure 6: Results for a fixed blocking factor of $B = 200$ for different square matrices ($N = m$).

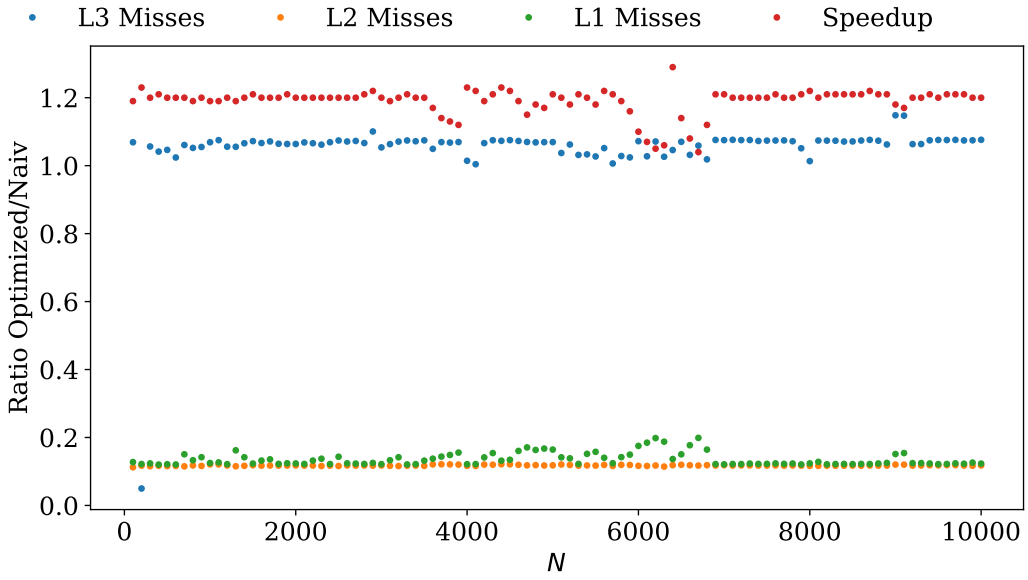

Figure 7: Results for a fixed blocking factor of $B = 200$ and fixed $M = 10000$ for different N.

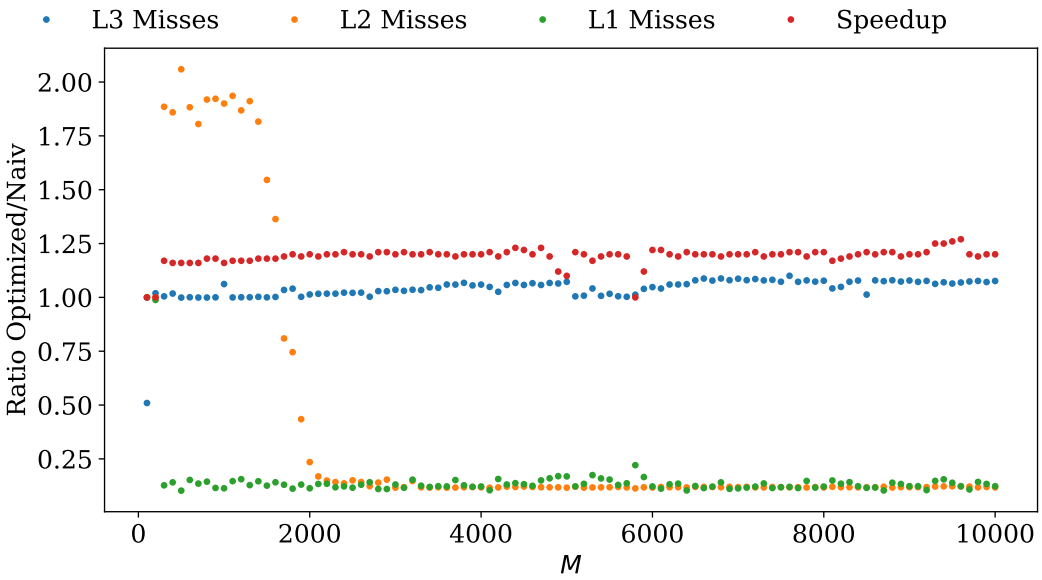

Figure 8: Results for a fixed blocking factor of $B = 200$ and fixed $N = 10000$ for different M.