Exercise 2: Performance Benchmarks, Finite Difference Discretization - a Report

Paul Kang, Bailiiang Li, Max Spannring, Moritz Techen

November 2024

Introduction

The following is a report, discussing the results we obtained for the three tasks of exercise 2, as a part of the lecture 360.242 Numerical Simulation and Scientific Computing I.

1 Task 1: Benchmark Vector Triad

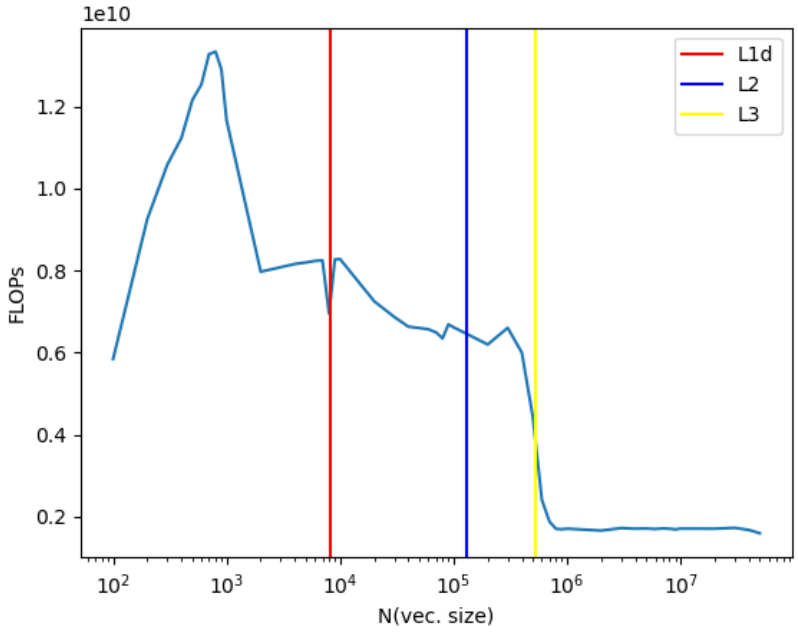

Figure 1: FLOPs vs. Vector size of the Benchmark. Cache levels are indicated by three vertical lines, separately.

The first task describes the process of benchmarking a CPU using a vector algebra algorithm that iterates addition and element-wise vector multiplication:

Algorithm 1 pseudocode

1: **for** $i = 1$ to N **do**
2: $\quad A[i] = B[i] + C[i] * D[i]$
3: **end for**

The processor computes two floating-point operations per iteration. When looping through the vector size N, the program measures the computation time to calculate the FLOPs number.

Cache and memory: As shown in figure 2, the memory space allocated for vectors of different sizes (N) illustrates the space of cache levels. We observe that as the length of a vector increases, the space occupied by the vector becomes larger and larger. So at the point of $N \approx 8000$, our 256 KiB L1d cache is completely filled, and all vectors bigger than this must be allocated in the L2

cache. The intersection between the red line and the FLOPs performance is the switching point between L1d and L2 levels.

Similarly, for the L2 cache of 4 MiB, the switching point from L2 into L3 corresponds to $N \approx 125,000$. We can prove this correspondence as an example, analytically. The algorithm with vector size $N = 125,000$ will loop 125,000 times. The algorithm allocates 4 vectors (A, B, C, and D) and each vector has 8 bytes per element (float64). In total, the memory occupation is:

$$4MiB = 4Vectors \times 8Bytes \times 125,000$$

which is the same as our cache size.

At $N = 500,000$, there is a switch from L3 cache to RAM. And CPU performance has remained steady since then.

CPU performance: In figure 2 L1d level, the FLOPs reach the peak performance of approximately 14 GFLOPs with a vector of around 1000 elements ($N \approx 1000$). After that, CPU performance in L1d drops sharply to approximately 8 GFLOPs. The FLOPs keep a general tendency of decreasing through L2 and L3 levels, until when $N > 500,000$, where vectors are saved in RAM.

The above observation is because the bandwidth of RAM is drastically slower than that of the cache. The performance is limited by how fast the CPU can read data from and write data into the RAM, and this limitation is called memory bound. For smaller vector sizes, we can see that the computational power of the CPU predominates. However, the performance is quickly dominated by the memory and bandwidth limit of caches as vector size increases. In the L1d level, the FLOPs drop after the peak can be interpreted due to the fact of reusing already cached numbers, so the CPU only needs to calculate and write the output.

Hardware: All the above results may vary on different processors. Our results are based on an AMD Ryzen 7 3700X with 8 cores, 256 KB L1d cache, 4 MB L2 cache and 16 MB L3 cache (in wsl).