Michigan State University

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

Homework 3

Alexander Harnisch

---

1) Compilers and auto-vectorization

a.

The Intel16 system's CPU used here is a Xeon E6-2680 by Intel, running at a clock speed of 2.4 GHz. Figure 1 shows the cache size and layout. However, the processor supports turbo boosting at 3.3 GHz for a single core and 2.9 GHz for all 14 cores at once. The CPU has 14 physical cores with Intel's hyper-threading technology, effectively having 24 virtual cores or simultaneous threads. Assuming the processor is capable of one floating point operation per clock cycle and thread running at 2.9 GHz turbo boost, the theoretical peak performance would be 96.6 GFLOP/s. However, the assignment is not precise enough to know what exactly it is asking for. Single thread performance? Turbo boost yes or no? The theoretical single thread peak performance at a boosted clock frequency of 3.3 GHz would simply be 3.3 GFLOP/s. The sustainable single thread peak performance at 2.4 GHz would be 2.4 GFLOP/s. Multiply that by 24 for sustainable multithread peak performance.

Cache Info [Edit Values]

L1I$	448 KiB	14x32 KiB 8-way set associative (per core, write-back)
L1D$	448 KiB	14x32 KiB 8-way set associative (per core, write-back)
L2$	3.5 MiB	14x256 KiB 8-way set associative (per core, write-back)
L3$	35 MiB	14x2.5 MiB 20-way set associative (shared, per core, write-back)

Figure 1: Xeon E5-2680 v4 cache size and layout [1].

b.

This depends on many assumptions that have to be made and are not given by the problem statement. Assuming 32 bit single-precision floating point numbers, AVX can perform the same instructions on up to 8 data streams at a time. Four at a time for double precision. That would mean we get another factor of 8 or 4. Additionally, assuming that all operations are of the type $a + b \times c$ which can be done in one operation using FMA we get another factor of 2.

c.

The plot using GCC is given by Figure 2. Figure 3 shows the results using the Intel compiler.

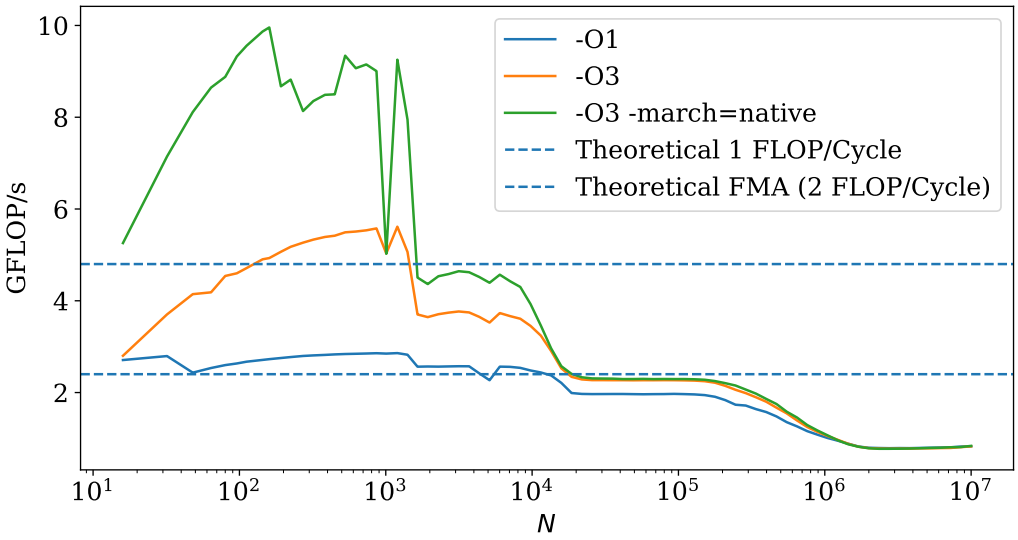

Figure 2: Performance of the vector triad kernel using different compiler settings for the gcc compiler.

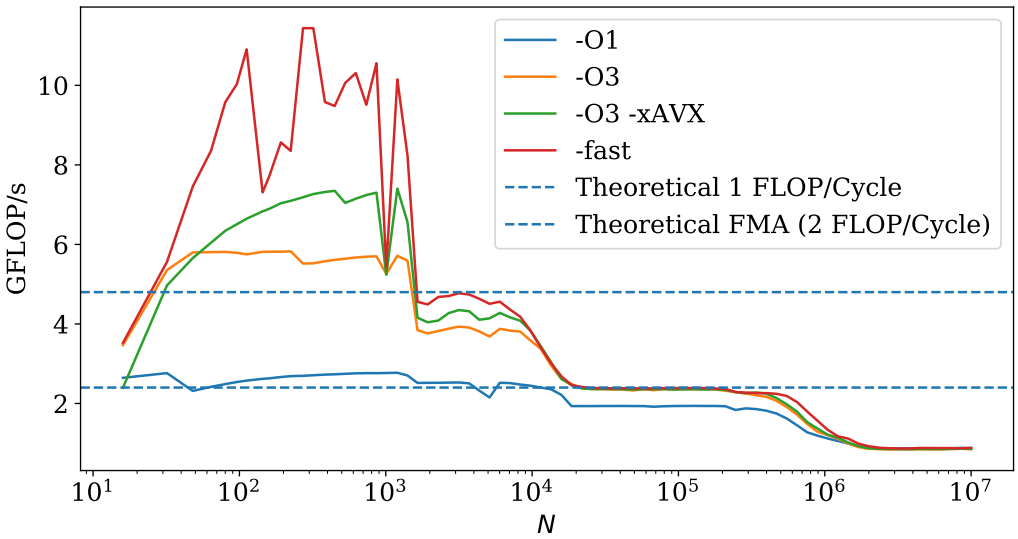

Figure 3: Performance of the vector triad kernel using different compiler settings for the Intel compiler.

d.

It seems like the O3 flag enables FMA. With only O1 the measured performance more or less consistently hovers around the expected peak performance assuming 1 FLOP per cycle. For small N that make perfect use of the L1 cache, the results are inconclusive. They do not reach the theoretical peak performance.

e.

There is a clear step structure in both plots, caused by the cache structure. The steps occur exactly where the corresponding cache level is used up according to Figure 1, since $N \cdot 8$ byte is roughly the amount of fast memory used.

2) Stream benchmark and memory bandwidth

a.

The bandwidths reported by STREAM are shown in Table 1.

Function	Best Rate MB/s	Avg time	Min time	Max time
Copy:	10718.2	0.150111	0.149278	0.152882
Scale:	11116.3	0.144151	0.143933	0.144601
Add:	11912.0	0.202684	0.201477	0.209135
Triad:	11863.3	0.202537	0.202304	0.202677

Table 1: STREAM output for $N = 10^8$.

b.

The memory bandwidth varies depending on the compiler and settings. See next section.

c.

The results are shown in Figure 4 and Figure 5.

d.

We can barely measure the bandwidth for small N, where the L1 cache is big enough the hold all data. In the L2 are, the STREAM result and gcc result are in agreement. However, for larger N the simple vector copy code underperforms.

For some reason the time measurement seems to be less accurate when using the Intel compiler, even though for both figures I did use the exact same source code. It seems like for small N the time is just so small that it can not be measured accurately.

I'm sad that the results are inconclusive, I put a lot of time into it but just don't have more time for this assignment.

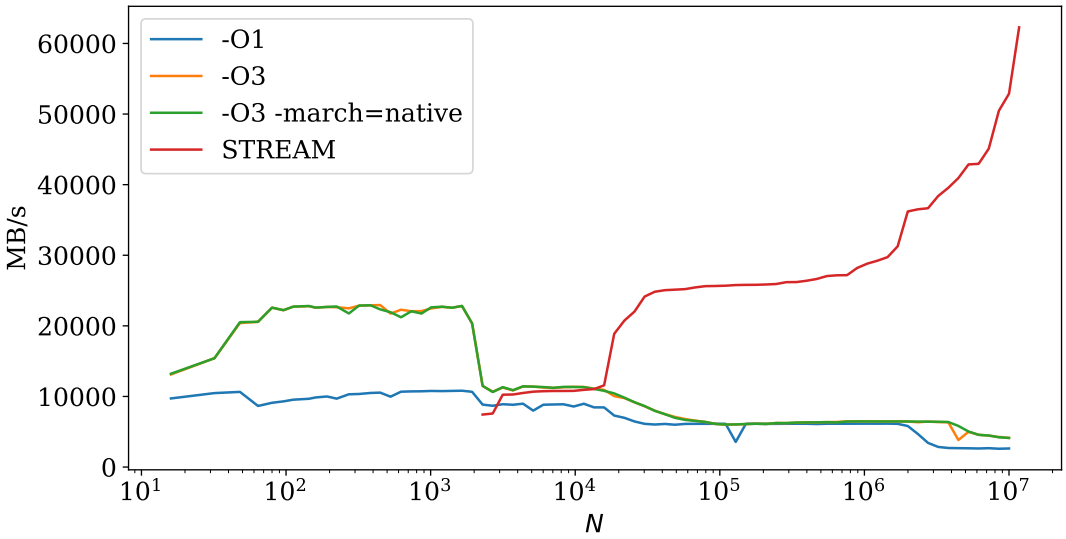

Figure 4: Results using the gcc compiler.

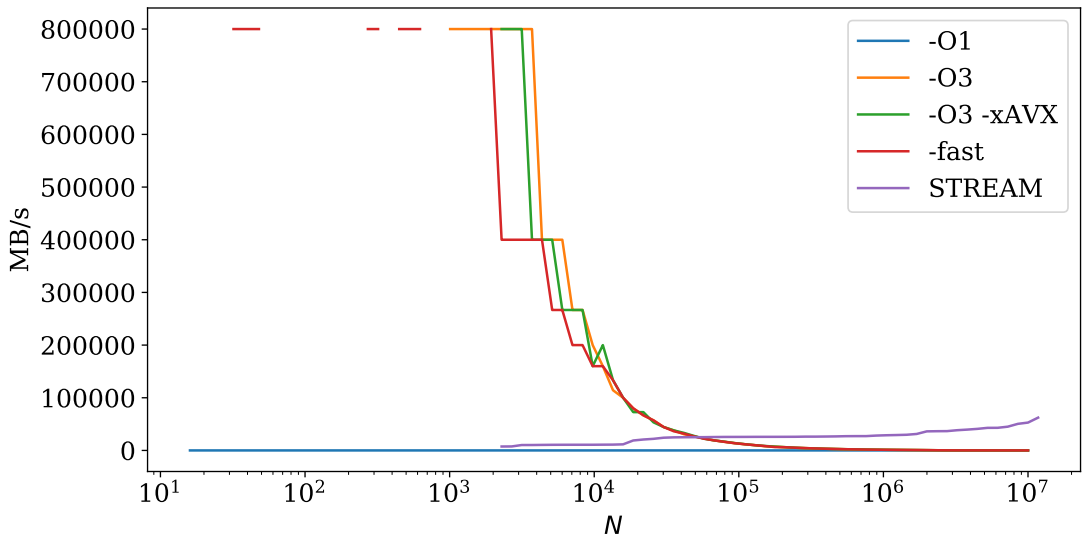

Figure 5: Results using the Intel compiler.

References

[1] WikiChip. *Xeon E5-2680 v4 - Intel*. https://en.wikichip.org/wiki/intel/xeon_e5/e5-2680_v4.