Boyao Zhu

CMSE 822

HW 3

Due Oct 7

1.

(a)

L1d cache	L1i cache	L2 cache	L3 cache	CPU(s)	CPU MHz
32K	32K	256K	35840K	28	2899.951

Asssuming the processor is capable of one floating point operation per clock cycle, the theoretical peak performance of this system is

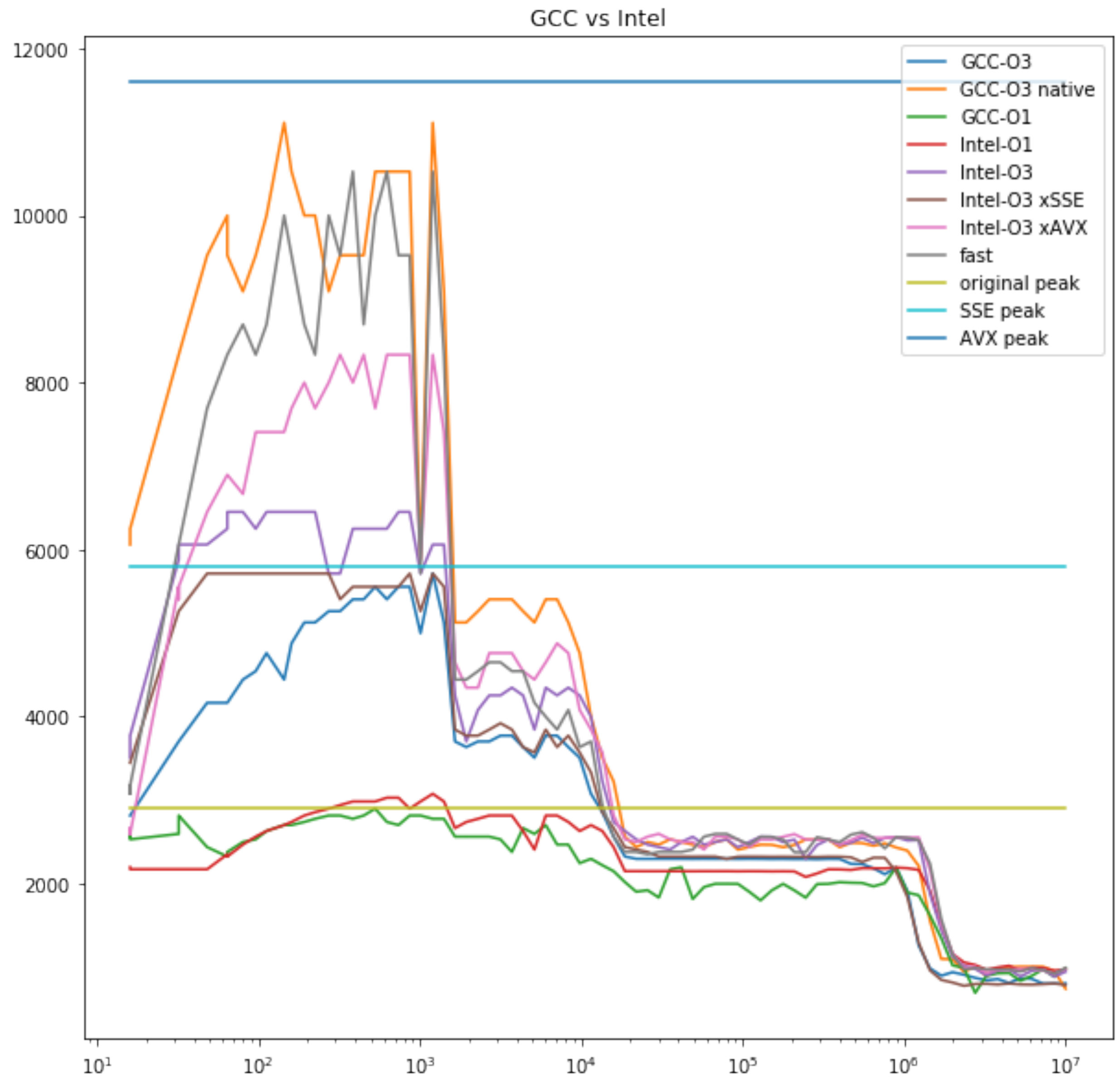
$$Performance == 2.9\ GFlop/s$$

(b)

Vectorization is the process of converting an algorithm from operating on a single value at a time to operating on a set of values at one time. So vectorization will improve CPU performance.
For SSE, the vector size is 128 bits (32 bytes, or 4 double variables), Therefore, the theoretical peak performance with SSE will be $2 * 2.9 GFlop/s = 5.8\ GFlop/s$.
For AVX, the vector size is 256 bits, so the peak performance is $11.6\ GFlop/s$.
For FMA, multiply and addition are fused. SO with FMA(without vectorization) only, the peak performance will be $2 * 2.9\ GFlop/s = 5.8\ GFlop/s$. For FMA with SSE, the peak performance is $4 * 2.9\ GFlop/s = 11.6\ GFlop/s$. For FMA with AVX, the peak performance is $2 * 4 * 2.9 GFlop/s = 23.2\ GFlop/s$.

(c)

The figures of performance of the vector triad kernel at different vector length N are shown below. I plot GCC compiler and Intel compiler in the same diagram, with labels. In addition, the theoretical peak performance estimations also are also plotted, three horizontal lines.



(d)

Comparing the performance between two compilers, we can see that with similar options, Intel compiler outperforms GCC compiler. More CPU's features for optimization should be used by Intel compiler.

For GCC, the peak performance of O1 and O3 are above the theoretical peak performance with no optimization (the lowest horizontal line) but below the theoretical estimation for SSE (the middle horizontal line. Thus GCC's O1 and O3 options may enable SSE.

For GCC's "-O3 -march=native" option, the peak performance is above the middle horizontal line, but below the theoretical estimation for AVX. Thus this option may enable AVX or FMA+SSE

(e)

For small vector length N, the performance does not reach the peak. The main reason should be that for small N, it does not take much time. When N increase roughly to 1000~2000, there is a sudden drop of performance. At that point, the memory consumption in this kernel is about 24~48 KB, while L1 data cache, as shown in (a) is 32 KB. So this drop is caused by the exhaustion of L1 data cache. Similarly, there are other drops when N~10000 and N~10000000, which corresponds to the exhaustion of L2 cache and L3 cache, respectively. i.e. when N~10000, the memory consumption is about 240 KB, while the L2 cache is 256 KB. When N~1000000, the memory consumption is about 24 MB, while the L3 cache is 35 MB.

2.

(a)

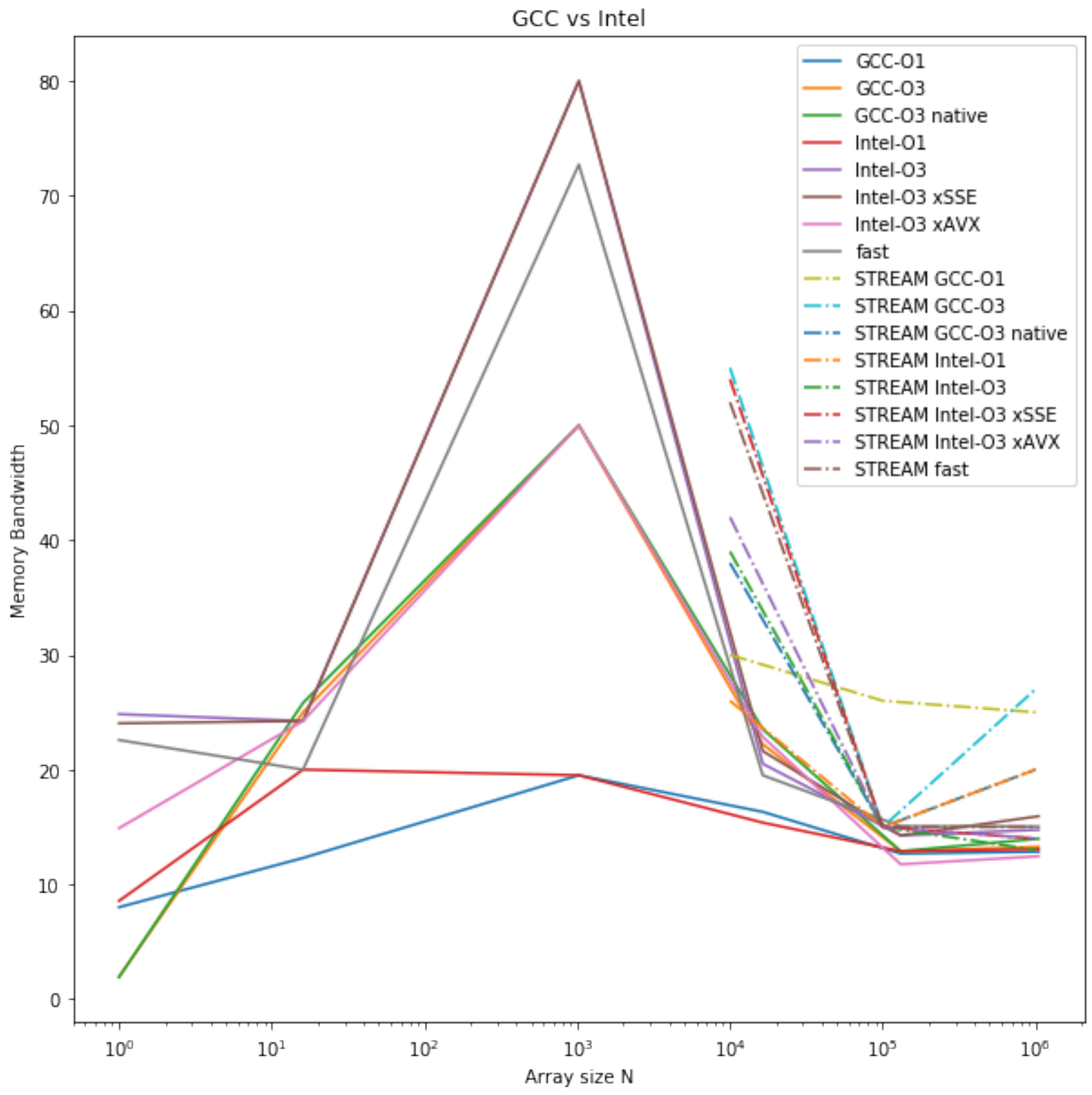
The memory bandwidth by STREAM for Intel16 at HPCC is 10895.4 MB/s (Copy), 10908.1 MB/s (Scale), 11590.9 MB/s (Add), 11908.3 MB/s (Triad). The array size is 10000000. The memory requirement of each array is about 75 MB, which is larger than twice the size of cache. NTIMES = 10 means each kernel is executed 10 times.

(b)

For vector length n = 2000000, the memory bandwidth is 9.8 GB/s.

To compare with part a, my result is lower than STREAM's. Cache can impact the bandwidth and how the compiler optimizes can also influence the performance. In this case, "gcc -O2" was employed. I also used "gcc -O3", which results in larger bandwidth and quite closer to STREAM's.

(c)



The figures above was calculated at different vector length N. The results from STREAM are shown by dashed lines. The compilers and corresponding options are given in the legend. When N < 10000, STREAM can not give an effective number, so the results of STREAM start from N >= 10000.

Comparing two compilers with different options, we can see that with similar options Intel compiler give better bandwidth than GCC compiler. However, when N is large, like N >

1000000, different compilers and options give similar memory bandwidths, which means that for large N the performance is mainly determined by memory bandwidth.

(d)

When N is small, the bandwidth is small, because instructions don’t take much time.
When N is between 100 ~ 10000, the bandwidth is much higher than what it should be.
When N is between 10000 ~ 100000, there is a drop of bandwidth of performance, which result from the exhaustion of L2 cache.
When N is larger than 1000000, which are not shown here, there should be another drop of bandwidth performance due to the exhaustion of L3 cache.
So we can estimate the bandwidth of L2 cache is roughly 80 GB/s. This estimation is smaller than the true bandwidth of caches because memory bandwidth has some impact on the performance when N is small.