ASSIGNMENT 1: Using the CPU and the GPU

In this assignment, we run prepared code on cluster CPU, GPU and on CPU of own laptop.

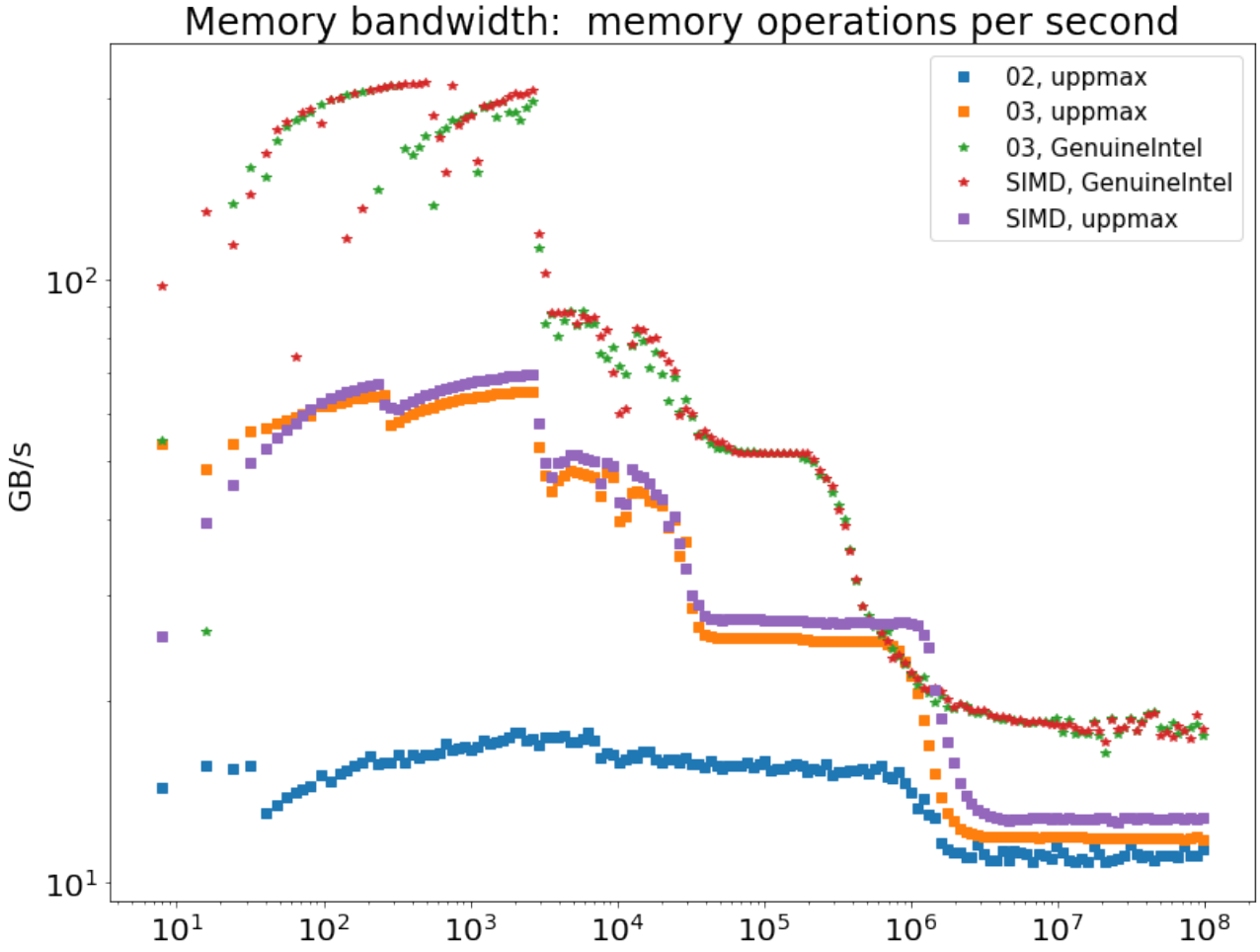
The main point of code is calculation of array elements:

```
void compute_triad(const std::size_t N, const float  a, const float *x, const float *y, float *z)
{
  for (std::size_t i = 0; i < N; ++i)
    z[i] = a * x[i] + y[i];
}
```

Main metrics:

- Memory bandwidth: memory operations per second (Gb/s)
- Updates per time (MUPD/s)

CPU performance



Above plot illustrates the dependence of memory performance on vector size.

As we can see, the program which was compiled with -O2 flag optimization has the worst memory performance. Optimization flag -O3 allows to include Vectorization.

The performance of vectorized program and SIMD-version (which has own class for vectorized Array) has approximately the same performance. Data-level parallelism technique allows to get more performance.

GPU performance

Here we use the same program for cuda, implemented for floats operations and double operations. For quite a large vectors (~10^7 size) cuda allows to achieve more memory performance in comparison to CPU usage.