1 Report of Homework 3

In homework 3 I implemented FFT using 3 techniques:simple serial program, parallelization using OpenMP, and parallelization using Nvidia CUDA(on GPU), and then compared the performance of them. I found the CUDA version is the most efficient when the I/O latency between the CPU/GPU and the system RAM is NOT a bottleneck (such bottleneck happens when N is small, hence most time is wasted on latency). Under this circumstance, the GPU version is around 10-200 times faster than the serial version and the OpenMP version is around 2-3 times faster than the serial version. (Note: My CPU has 4 cores and my GPU has 768 CUDA cores with 4GB onboard VRAM.) I tested N up to 26 where the complex double list takes up $2 * 8Bytes * 2^{16} = 1GB$ of memory. The results of the OpenMP and GPU version are shown in figure 1 and 2.

```
FFT_OPENMP
  C/OpenMP version

  Demonstrate an implementation of the Fast Fourier Transform
  of a complex data vector, using OpenMP for parallel execution.


  Accuracy check:

    FFT ( FFT ( X(1:N) ) ) == N * X(1:N)

           N      NITS    Error         Time          Time/Call     MFLOPS

           2      1000  7.859082e-017  6.299996e-002  3.149998e-005      0.317461
           4      1000  1.209837e-016  1.090000e-001  5.449998e-005      0.733945
           8      1000  6.820795e-017  1.880002e-001  9.400010e-005      1.276594
          16      1000  1.438671e-016  2.349999e-001  1.174999e-004      2.723405
          32       100  1.331210e-016  3.099990e-002  1.549995e-004      5.161307
          64       100  1.776545e-016  3.200006e-002  1.600003e-004     11.999976
         128       100  1.929043e-016  3.099990e-002  1.549995e-004     28.903320
         256       100  2.092319e-016  6.299996e-002  3.149998e-004     32.507955
         512        10  1.927488e-016  0.000000e+000  0.000000e+000      1.#INF00
        1024        10  2.310923e-016  0.000000e+000  0.000000e+000      1.#INF00
        2048        10  2.447624e-016  1.600003e-002  8.000016e-004    140.799715
        4096        10  2.486633e-016  1.500010e-002  7.500052e-004    327.677708
        8192         1  2.581515e-016  0.000000e+000  0.000000e+000      1.#INF00
       16384         1  2.729841e-016  0.000000e+000  0.000000e+000      1.#INF00
       32768         1  2.922632e-016  0.000000e+000  0.000000e+000      1.#INF00
       65536         1  2.834224e-016  0.000000e+000  0.000000e+000      1.#INF00
      131072         1  3.150036e-016  0.000000e+000  0.000000e+000      1.#INF00
      262144         1  3.220390e-016  3.100014e-002  1.550007e-002   1522.119709
      524288         1  3.280359e-016  4.699993e-002  2.349997e-002   2119.465224
     1048576         1  3.285315e-016  9.299994e-002  4.649997e-002   2255.003723
     2097152         1  3.509965e-016  1.730001e-001  8.650005e-002   2545.674407
     4194304         1  3.559205e-016  4.470000e-001  2.235000e-001   2064.310570
     8388608         1  3.711943e-016  8.340001e-001  4.170001e-001   2313.404777
    16777216         1  3.656319e-016  1.755000e+000  8.775001e-001   2294.319987

FFT_OPENMP:
  Normal end of execution.
```

Figure 1: OpenMP version.

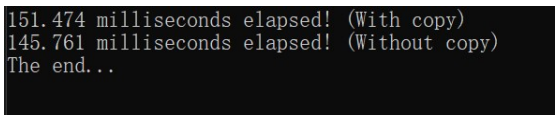

Figure 2: CUDA version.