Lab4 Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | serial | cuda\_arbitrary\_64 | cuda\_simd\_64 | cuda\_simd\_128 |
| 100 | real 0m23.280s user 0m23.210s sys 0m0.005s | real 12m50.065s  user 9m8.879s  sys 3m41.072s | real 11m52.915s  user 8m24.775s  sys 3m27.835s | real 6m39.084s  user 4m48.288s  sys 1m50.606s |
| 1000 | real 3m35.259s user 3m35.208s sys 0m0.010s | real 86m13.806s  user 61m8.849s  sys 25m4.260s | real 79m5.248s  user 55m12.704s  sys 23m51.921s | real 43m5.448s  user 29m40.891s  sys 13m24.193s |
| 5000 | real 18m23.693s user 18m23.558s sys 0m0.012s | more than 2 hr | more than 2 hr | more than 2 hr |
| 10000 | real 37m12.130 user 37m11.970 sys 0m0.026s | more than 2 hr | more than 2 hr | more than 2 hr |

In the lab4, we have two version: arbitrary and SIMD.

First, I find out that CUDA programs run much longer than the serial one. I am not pretty sure about the reason. But I guess that the first thing is read the data from the source file is serial not parallel. And maybe something changed in the transform function.

Second, when I increase the number of threads, there will be a great improvement in performance like

for SIMD PC\_data\_t00100 from 64 threads (12min) to 128 threads (6min) and for SIMD PC\_data\_t01000 from 64 threads (79min) to 128 threads (43min). It is nearly half of the time.

Third, when we use SIMD with the same number of threads, there will also be an improvement. And when the data is 100, the improvement is very tiny about 1min. When it comes to data size of 1000, there will be a 7 min promotion.

And due to the time limit, the program of 5000 and 10000 were forced to stop. We can see that it runs so long.