Assignment 5 , 159.735, 2020 S2

Family Name: JIANG , Given Name: TAO ,

Student ID:20004769

**Assignment5 :** **Quantum warp drive engine**

The experiment uses GPU to accelerate the calculation of spatial dimensions greater than 3. The experiment uses an Nvidia graphics card with 3G memory.

* **User’s Guide**

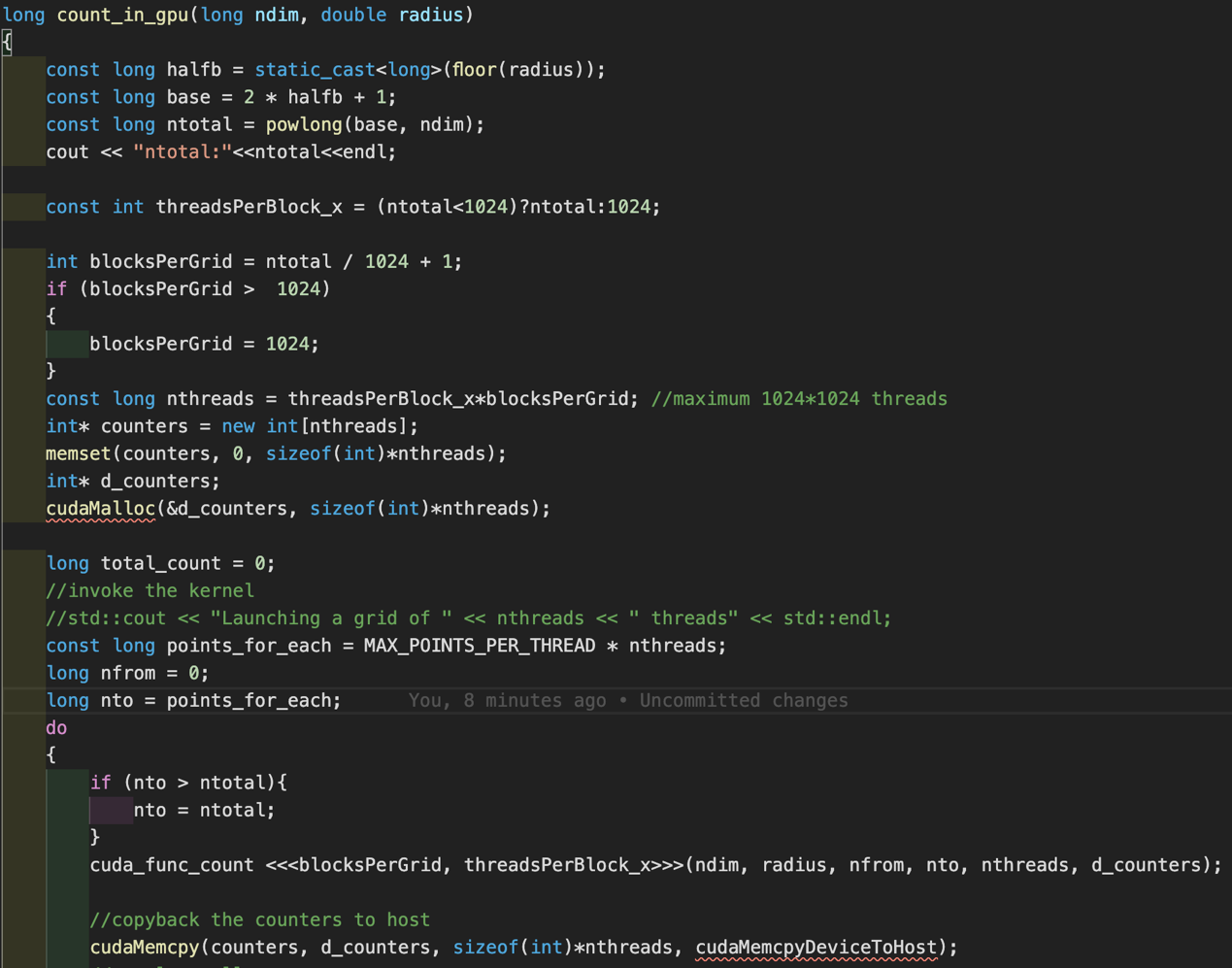
I use Ubuntu as a development platform.

Ubuntu env：

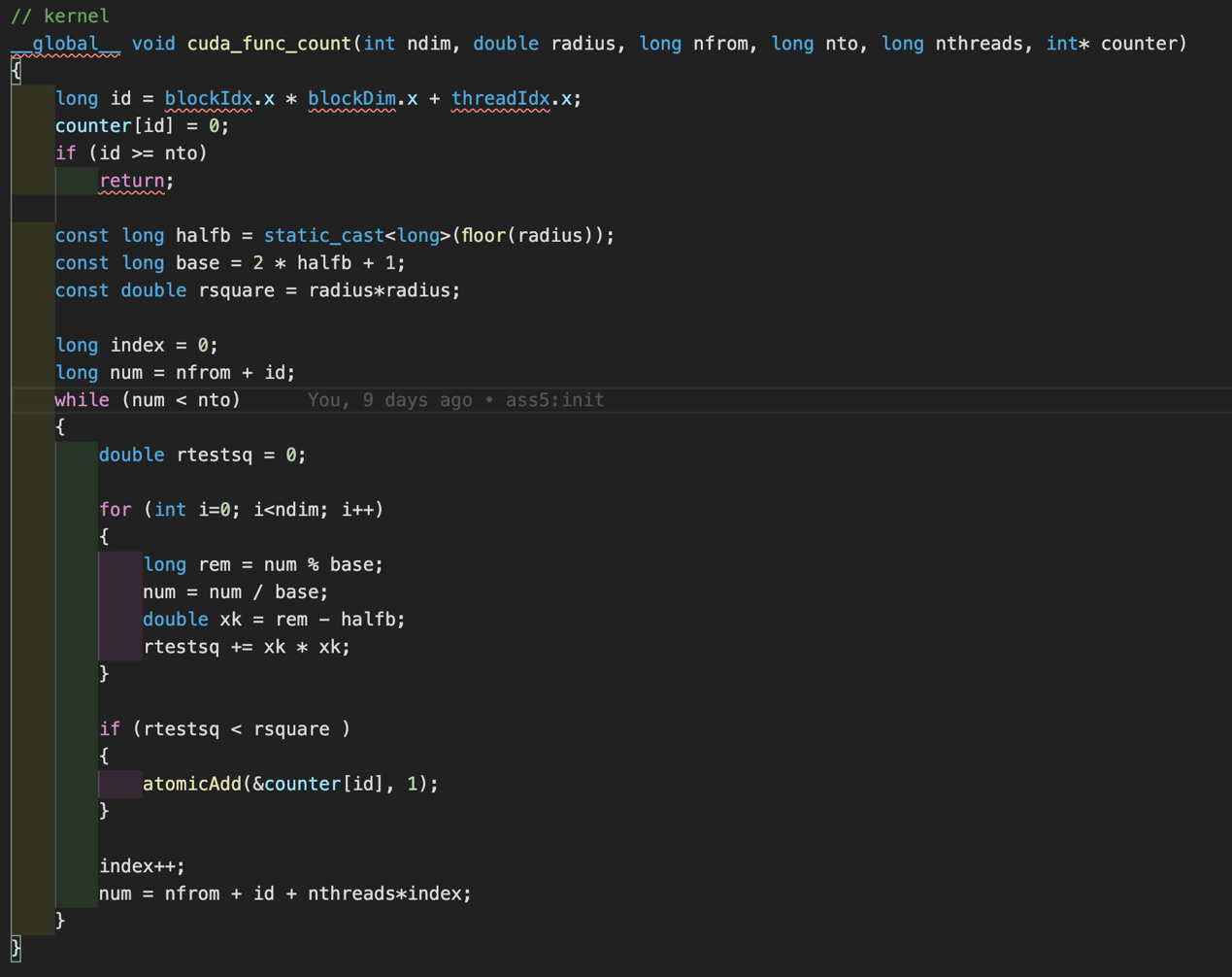
```shell  
nvcc -o nsphere\_cuda nsphere\_cuda.cu -std=c++11  
./ nsphere\_cuda   
```

* **Algorithm**

The algorithm of this experiment is based on *count\_in\_v1()* in the file **nsphere.cpp** to achieve GPU parallelism. In the *count\_in\_gpu()* function, the *ndim, radius, nfrom, nto, nthreads, d\_counters* parameters are sent to the GPU for calculation through *cuda\_func\_count()*.



In the GPU, convert the decimal number into another base system. In order to obtain the key parameter of *rtestsq*, and compare with *rsquare* to see whether the point meets the requirements.



Copy the *d\_counters* on the GPU to the CPU for the final statistics.

* **Experiment results**

The experiment uses random radius and random dimensions to test CUDA parallel programs and sequential programs. The running time of both is recorded in ms. At the same time, the speedup of the CUDA program is recorded. The results are as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **r** | **nd** | **ntotal** | **count** | **cuda Time** | **seq Time** | **speedup** |
| 2 | 5 | 3125 | 221 | 54.117 | 1.207 | 0.022303528 |
| 2.24979 | 5 | 3125 | 333 | 0.441 | 0.763 | 1.73015873 |
| 4.18761 | 6 | 531441 | 27273 | 5.115 | 151.348 | 29.58905181 |
| 2.55379 | 7 | 78125 | 3081 | 1.131 | 25.614 | 22.64721485 |
| 5.1605 | 7 | 19487171 | 446939 | 28.761 | 5412.98 | 188.2055561 |
| 3.39907 | 1 | 7 | 7 | 0.394 | 0.021 | 0.053299492 |
| 7.59039 | 1 | 15 | 15 | 0.374 | 0.008 | 0.021390374 |
| 5.33657 | 3 | 1331 | 619 | 0.383 | 0.181 | 0.472584856 |
| 6.60231 | 3 | 2197 | 1213 | 0.383 | 0.295 | 0.770234987 |
| 3.51416 | 2 | 49 | 37 | 0.37 | 0.012 | 0.032432432 |
| 7.25589 | 5 | 759375 | 104767 | 5.596 | 150.863 | 26.95907791 |
| 7.52157 | 7 | 170859375 | 6467039 | 198.467 | 46945.4 | 236.5400797 |
| 6.86258 | 7 | 62748517 | 3499165 | 74.891 | 17124.3 | 228.6563138 |
| 7.31789 | 4 | 50625 | 13921 | 0.659 | 8.379 | 12.71471927 |
| 2.46065 | 6 | 15625 | 1341 | 0.443 | 3.726 | 8.410835214 |
| 7.90935 | 2 | 225 | 193 | 0.386 | 0.026 | 0.067357513 |
| 7.36344 | 4 | 50625 | 14881 | 0.638 | 8.37 | 13.11912226 |
| 2.6055 | 4 | 625 | 233 | 0.387 | 0.125 | 0.322997416 |
| 2.11905 | 6 | 15625 | 485 | 0.434 | 3.68 | 8.479262673 |
| 6.07326 | 1 | 13 | 13 | 0.405 | 0.007 | 0.017283951 |

* **Conclusion**

We can see from table1 that the parallel computing speed of CUDA is much higher than sequential in high dimensionality. GPU's multi-threads play an essential role.

However, when the dimension is lower than 3, the calculation speed of CUDA is not faster than sequential. The reason is the amount of calculation is small, and the communication between the CPU and the GPU has certain consumption. It makes CUDA lose the advantages of parallel computing.

I also found a phenomenon that the speed of CUDA parallel running for the first time is very slow. The sequential program takes 1.2ms, but the CUDA program takes 44ms. This GPU is in the state of saving points in the non-studio, and it takes more time to start up for the first time. So the above phenomenon occurs. Therefore, the GPU should be warmed up when testing.

It can be seen from the above chart1 that the sequential program execution time speed tends to increase as the lenses increase. However, the GPU parallel computing speed is very fast and stable. The reason is there are many threads on the GPU, and the number of parallel tasks does not exceed this value, so each thread is executed quickly.

The Chart2 is the GPU speed-up when calculating different lenses. More threads can speed up even more. The results of this experiment conform to Gustafson's Law.