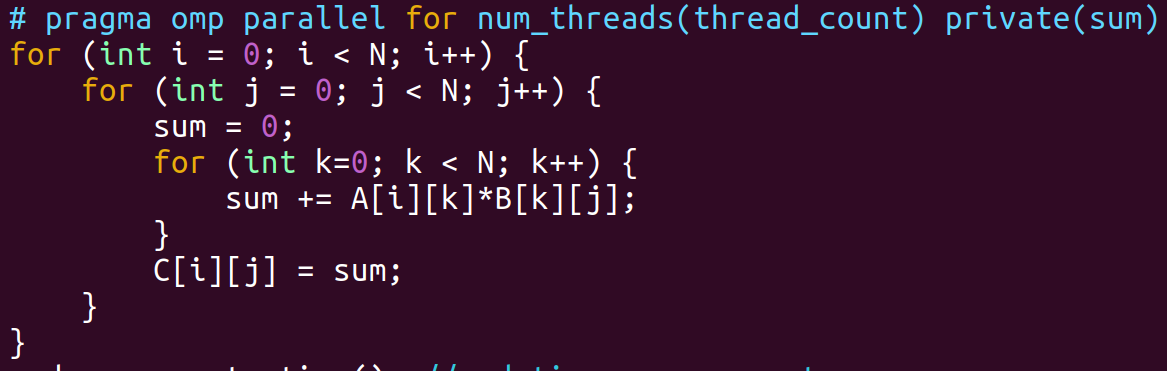
# Problem 1

The two experiments both run in a 64-bit Ubuntu virtual machine with Core: 4 and Memory: 8GB.

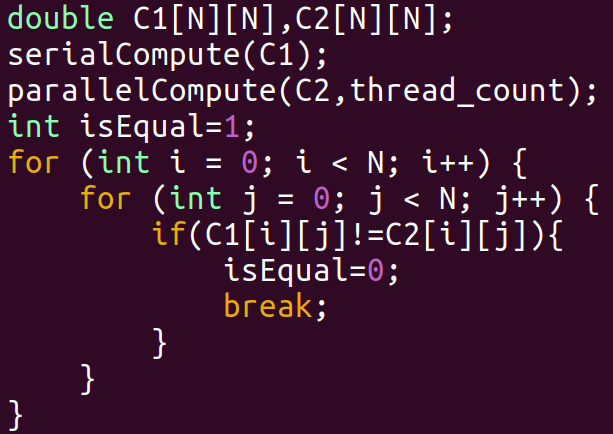
## Experimental Ideas

Matrix multiplication mainly consists of three loops, and omp pragma can be added before each loop. After verification, the outermost loop can achieve the highest degree of parallelism. The code is as follows:

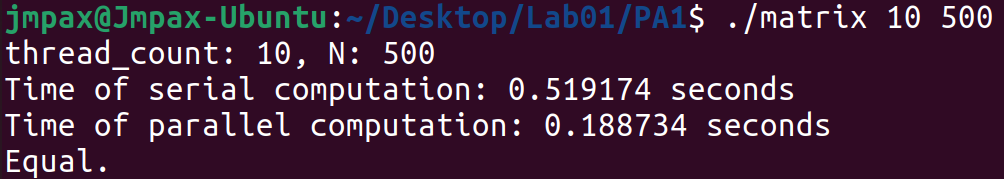


In addition, I also tested the row-first matrix multiplication algorithm, but the effect was not as good as the basic algorithm, because in the priority algorithm, pragma can only be added to the middle loop, and every time the outer loop variable is updated, the thread will wait.

I used functional programming to perform serial compute and parallel compute and validate each element of the resulting matrix.



## Result Verification

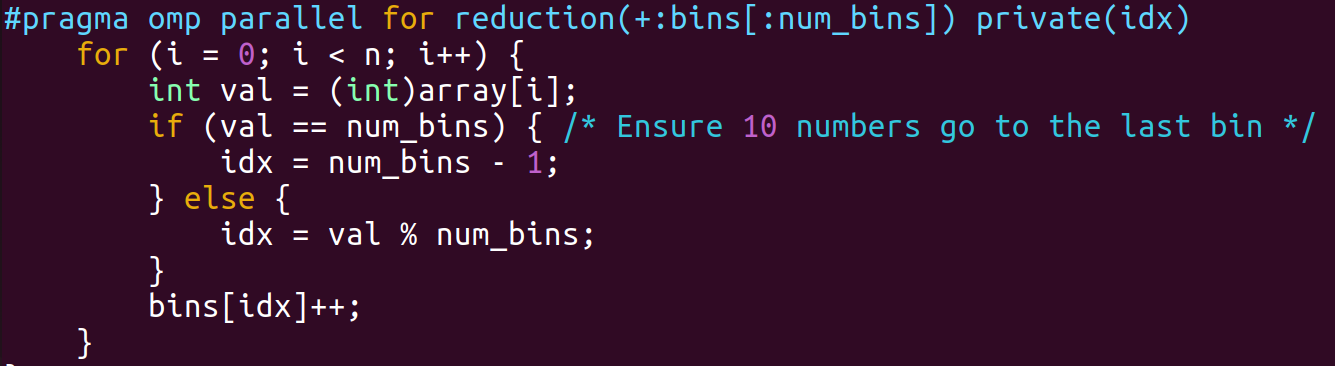


As you can see, we can get about 2.75 times improvement by using 10 thread concurrency on a 500-dimensional matrix.

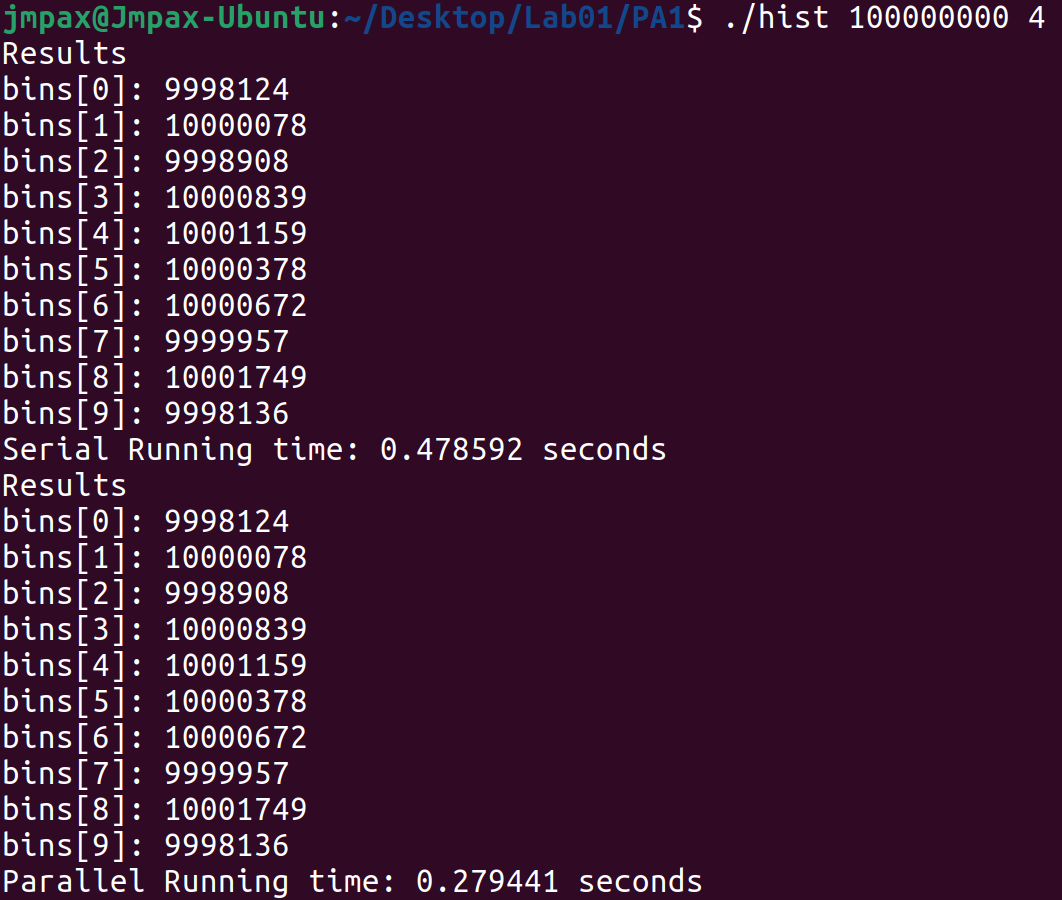
# Problem 2

## Experimental Ideas

The problem that needs to be solved in this experiment is the concurrent operation of multi-threaded histogram result statistical arrays. We can use the reduction operation provided by OpenMP to let each thread process the results of processing each part and finally combine them into a global variable. The code is as follows:



## Result Verification



It can be seen that at the data level of , the concurrency of 4 threads has been improved by about 1.71 times. And the results of histogram statistics are consistent.