Report

Hanchao Chen

1, int num\_neighbours (int ii, int jj)

Since the counting 8 neighbors is an independent work, we can do it at the same time.

One way is to use omp parallel for.

The other way is to use omp sections: I made some changes on the original function, and crested a new “int num\_neighbours0(int ii, int jj)”. I divided the for loop into 8 sections and each one runs on one thread.

2, The nested for loop

I find it very tricky when dealing with the nested for loop. When I use two “#pragma omp parallel for” and turn on the “Nest(1)”, I find the code runs slower than before. I think it is because that there are N threads in the outer loop, while there are N\*N threads created in the inner loop. So that it causes much time to switch the threads and then make the code slower.

I also leave a nested parallel version named “void do\_iteration0(void)”, but it won’t speed up the code in my computer. For the “void do\_iteration0(void)” version, I only remain the omp parallel for the inner for loop, it can speed up the code a little.