P. Hadjidoukas

Set 4 - OpenMP II

Issued: March 29, 2023

Question 1: OpenMP bug hunting

Identify and explain any bugs in the following OpenMP code. Propose a solution. Assume all headers are included correctly.

```
1  #define N 1000
2
3  extern struct data member[N]; // array of structures, defined elsewhere
4  extern int is_good(int i); // returns 1 if member[i] is "good", 0 otherwise
5
6  int good_members[N];
7  int pos=0;
8
9  void find_good_members( )
10 {
11     #pragma omp parallel for
12     for(int i=0; i<N; i++) {
13         if (is_good(i)) {
14             good_members[pos] = i;
15
16             #pragma omp atomic
17             pos++;
18         }
19     }
20 }
```

Hints:

- Identify the race condition (as we saw in the class)
- In your solution you can use "omp critical" or "omp atomic capture" [1]

[1] omp atomic capture: OpenMP specs 3.1, section 2.8.5, especially page 74, lines 8-13

Question 2: Statistics

In `statistics.c`, the sequential diagnostics function `compute_max_density()` finds and prints the maximum density value and its location.

```
1  void compute_max_density(double *rho_, int N)
2  {
3      // rho_ : matrix of size NxN, allocated as one dimensional array.
4      // rho[i*N+j] corresponds to rho[i, j]
5      // This routine finds the value of max density (max_rho) and its
6      // location (max_i, max_j ) - it assumes there are no duplicate values
7
8      double max_rho;
9      int max_i,max_j;
10
11     max_rho=rho_[0];
12     max_i=0;
13     max_j=0;
14
15     for (int i=0 ; i<N; ++i)
16     for (int j=0 ; j<N; ++j)
17     {
18         if (rho_[i*N+j]>max_rho)
19         {
20             max_rho=rho_[i*N+j];
21             max_i=i;
22             max_j=j;
23         }
24     }
25
26     printf("=============================================\n" );
27     printf("Output of compute_max_density( ):\ n" );
28     printf("Max rho: %.16f\n", max_rho);
29     printf("Matrix location: %d %d\n", max_i, max_j);
30 }
```

Provide, in the function `compute_max_density_omp()`, a parallel OpenMP implementation of the previous code.

- Try to keep the number of memory accesses close to that of the sequential version.
- Study the hands-on example `find_max` of the last lecture (OpenMP part 2).