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.

```c
#define N 1000

extern struct data member[N]; // array of structures, defined elsewhere
extern int is_good(int i); // returns 1 if member[i] is "good", 0 otherwise

int good_members[N];
int pos=0;

void find_good_members( )
{
    #pragma omp parallel for
    for(int i=0; i<N; i++) {
        if (is_good(i)) {
            good_members[pos] = i;

            #pragma omp atomic
            pos++;
        }
    }
}
```

Hints:

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

In order to avoid data races between different updates of global variable pos, the code puts the increment in an atomic construct. However, the code does not work, because there is a data race between the read of pos right before the atomic construct and the write of pos within the construct.

```c
int mypos ;
#pragma omp critical
{
    mypos = pos ;
    pos++;
}
good_members[mypos] = i;
```

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

```
int mypos;

#pragma omp atomic capture
mypos = pos++;

good_members[mypos] = i;
```

Question 2: Statistics

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

```c
void compute_max_density(double *rho_, int N)
{
    // rho_ : matrix of size NxN, allocated as one dimensional array.
    // rho[i*N+j] corresponds to rho[i, j]
    // This routine finds the value of max density (max_rho) and its
    // location (max_i, max_j ) -- it assumes there are no duplicate values

    double max_rho;
    int max_i,max_j;

    max_rho=rho_[0];
    max_i=0;
    max_j=0;

    for (int i=0 ; i<N; ++i)
    for (int j=0 ; j<N; ++j)
    {
        if (rho_[i*N+j]>max_rho)
        {
            max_rho=rho_[i*N+j];
            max_i=i;
            max_j=j;
        }
    }

    printf("========================================\n" );
    printf("Output of compute_max_density( ):\ n" );
    printf("Max rho: %.16f\n", max_rho);
    printf("Matrix location: %d %d\n", max_i, max_j);
}
```

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).

A possible solution code for this question is depicted below. Similarly to previously studied examples, it is beneficial if we spawn a parallel region and have each thread compute its private maximum density value and its location. Finally, each thread updates the global result within a critical section.

Note: use of the collapse and the nowait clauses are not required in the solution because this was not specified / requested in the question.

```c
double max_rho;
int max_i, max_j;

max_rho=rho_[0] ;
max_i=0;
max_j=0;

#pragma omp parallel
{
    double lmax_rho = rho_[0];
    int lmax_i=0;
    int lmax_j=0;

    #pragma omp for nowait collapse(2)
    for (int i=0; i<N; ++i)
    for (int j=0; j<N; ++j)
    {
        if (rho_[i*N+j]>lmax_rho)
        {
            lmax_rho=rho_[i*N+j];
            lmax_i=i;
            lmax_j=j ;
        }
    }
    #pragma omp critical
    {
        if (lmax_rho>max_rho)
        {
            max_rho=lmax_rho;
            max_i=lmax_i;
            max_j=lmax_j;
        }
    }
}
```