Technische Universität München
Institut für Informatik
PD Dr. Ralf-Peter Mundani
Philipp Samfass, M.Sc.

SoSe 2018

High Performance Computing - Programming Paradigms and Scalability

Exercise Sheet 5: Loop Dependencies, OpenMP

1 Loop Dependencies

Given is the following code fragment inside two nested loops:

```
1  for(int i = 3; i <= N; i++) {
2      for(int j = 1; j <= M; j++) {
3          A[i-3][j] = 2 * B[i][j+2];
4          C[i][j]   = D[i,j+1] + A[i][j];
5          B[i][j-1] = C[i][j];
6      }
7  }
```

a) Examine these statements and name all occurring dependencies (flow dependence, antidependence, output dependence)! Input dependencies may be neglected.

b) State for each dependency both distance and direction vector in order to check if it is a *loop-carried* or a *loop–independent* dependency.

c) Could the two nested loops be (partially) run in parallel? Justify your answer!

2 Parallel Computation of π

With

$$\phi(x) = \frac{1}{1+x^2} \qquad \text{and} \qquad \int \phi(x)dx = arctan(x)$$

one can compute π via the integration of $\phi(x)$ over $[0, 1]$. The following code fragment shows how to compute π sequentially, subdividing the unit interval into N stripes.

```
1     int      i, N;
2     double   h, x, sum, PI;
3
4     h = 1/N;
5     sum = 0.0;
6
7     for(i = 1 ; i <= N ; i++) {
8         x = h * (i - 0.5);
9         sum += 4.0 / (1.0 + x*x);
10    }
11
12    PI = h * sum;
```

Extend the program with valid OpenMP directives to compute π in parallel and think about sufficient synchronisation of the threads!

3 Parallel min-max-Search

To find the minimal and maximal elements `min` and `max`, resp., of a 3-dimensional integer array A of size $3 \times 1000 \times 1000$, the following sequential code is used:

```
1     int  i, j, k, min, max;
2
3     min = A[1][1][1];
4     max = A[1][1][1];
5
6     for(i=1; i <= 3; i++) {
7         for(j=1; j <= 1000; j++ ) {
8             for(k=1; k <= 1000; k++) {
9                 if(A[i][j][k] < min) {
10                    min = A[i][j][k];
11                }
12                if(A[i][j][k] > max) {
13                    max = A[i][j][k];
14                }
15            }
16        }
17    }
```

Extend the program with valid OpenMP directives for the parallelisation of **one** loop and – if necessary – think about sufficient synchronisation of the threads!