Technische Universität München
Chair of Scientific Computing
Dr. Ralf-Peter Mundani
Dr. Andreas Paul

Parallel Programming and HPC
Exercise Sheet 4: Loop Dependencies, OpenMP

14th June 2011

1 Matrix Transposition

The sequential transposition of a matrix A of size $N \times N$ works as follows:

```
for( i=1 ; i<=N ; i++ ) {
    for( j=i ; j<=N ; j++ ) {
        tmp = A[i][j];
        A[i][j] = A[j][i];
        A[j][i] = tmp;
    }
}
```

a) In order to compute A^T in parallel, one can choose between function, data, and competitive parallelism. Depending on the underlying hardware (shared or distributed memory) these approaches have different strong and weak aspects. Give an implementation idea for each approach and discuss your results.

b) Compute for a data parallel approach with column-wise data decomposition the speed-up depending on N and p (the amount of processes) only! Exchanging two local elements `A[i][j]` and `A[j][i]` takes T_{EX} time, transmitting one element from process p_i to process p_j takes $T_{COM} = 1.5 \cdot T_{EX}$ time. For simplification you can further assume, that one process transmits all elements of its domain (i.e. this includes also the redundant transmission from p_i to p_i itself).
What can you observe for the parallel efficiency?

2 Advanced Loop Dependencies

Dependencies throughout different iterations of loops are difficult to detect. Here, the so called distance and direction vector help to distinguish between loop-carried and loop- independent dependencies. Analyse the following code fragment and identify all occurring dependencies.

```
    for( i=1 ; i<N ; i++ ) {
        for( j=1 ; j<N ; j++ ) {
            for( k=1 ; k<N-3 ; k++ ) {
S1:             A[i+1][j+1][k] = A[i][j][k] + A[i][j+1][k+3];
            }
        }
    }
```

Are the above dependencies to be solved? Justify your solution!

3 Parallel Computation of π

With

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

one could 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.

```
int    i, N;
double  h, x, sum, PI;
h = 1/N;
sum = 0;
for( i=1 ; i<=N ; i++ ) {
    x = h * (i - 0.5);
    sum = sum + 4/(1 + x*x);
}
PI = h*sum;
```

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

4 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:

```
int   i, j, k, A, min, max;
min = A[1][1][1];
max = A[1][1][1];
for( i=1 ; i<=3 ; i++ ) {
    for( j=1 ; j<=1000 ; j++ ) {
        for( k=1 ; k<=1000 ; k++ ) {
            if( A[i][j][k] < min ) { min = A[i][j][k]; }
            if( A[i][j][k] > max ) { max = A[i][j][k]; }
        }
    }
}
```

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