Lecture Parallel Processing

Assignment 1: OpenMP – Stencil Code

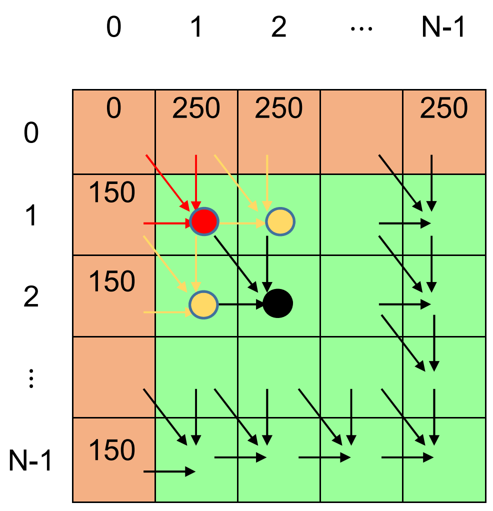
The assignment consists of 3 parts: development of a sequential scientific code, parallelization with OpenMP, and performance evaluation.

Part 1: Stencil Code and Sequential Program Version

Consider a square matrix $A \in \mathbb{R}^{N\times N}$ with $N = 46080$ and matrix elements represented as type **`float`**. The first column is initialized to **`150`**, the first row is initialized to **`250`**, and the first element of the matrix to **`0`**. For each element of the matrix except the left and upper border the following simple C stencil operation is performed:

$$\begin{aligned}A[i][j] = \big(&fabs(\sin(A[i-1][j-1])) && \ldots \textit{ diagonal element}\\ +\ &fabs(\sin(A[i][j-1])) && \ldots \textit{ left element}\\ +\ &fabs(\sin(A[i-1][j]))\big) && \ldots \textit{ upper element}\\ *\ &100\end{aligned}$$

Thus we get the following computational wavefront:



Write a sequential C or C++ program **`stencil`** which iterates over the matrix and calculates all elements. The sequential version shall be used to check the correctness of the parallel version. Program **`stencil`** is called with 6 integer values denoting three elements which shall be written to the standard output (used for testing), e.g.

`$ ./stencil 25 48 3000 2000 25673 41983` *writes the float values of elements*

`A[25][48]  A[3000][2000]  A[25673][41983]`

Part 2: OpenMP Parallelization

The goal is to develop a highly efficient parallel code version. The execution time of the program shall be improved by optimizing the OpenMP code.
During the experiments OS
commands shall be used to monitor the system.

Part 3: Performance Analysis and Report

Tasks to be done:

1. Develop an OpenMP version and compile it.

2. Determine the optimal number of threads and calculate the speed-up compared with the sequential version.
3. Examine the system during execution:
   How many processors are visible? How many threads are created? What about work balance and task scheduling?
4. Try to get better results by modifying the OpenMP source.

Write a short report document (mainly charts) covering the following topics (not more than 3 pages):

1. Explain your parallelization strategy.
2. Report your main OpenMP pragmas. Is your solution based on the C or C++ ?
3. Report the best execution times (show the number of threads, options, etc. mainly responsible for obtaining good results).
4. Show the speed-ups compared with the serial version.
5. Report about the OS commands which you have been using for monitoring the system and their usefulness and the behavior of the system you have noticed.

Execution Environment

Your laptop.

-

Requirements for positive grading

- Source files and report must have been submitted.
- Source files must compile/link correctly.
- Program **`stencil`**:
    - values must be computed correctly
    - performance measurements must have been done and documented
    - a substantial speedup must be achieved

Beware of the size

Dimension of 46.080x46.080 results in 2.123.366.400 elements and a matrix size of about 8.5 GB. A four-byte integer runs from -2.147.483.648 to 2.147.483.647.

Deadline

Deadline: 09.12.2019