Student projects

(Dec. 5, 2016 until Jan. 9, 2017)

- Work in teams of 4-6 students
- Deadline to submit projects: Jan 9, 2017
- Please submit the projects by email to your teaching fellow.
    - Group 1&2: Holger.Hennig@uni-rostock.de
    - Group 3: Markus.Wolfien2@uni-rostock.de
    - Group 4: Faiz.Khan2@uni-rostock.de
- Please send a single cpp-file per task specifying the names and student numbers of all participants of the group in the header. Document your C/C++ code in the cpp file with the comment function.
  In addition, please also submit a short summary (1-2 pages maximum) about the project, e.g., describing matrix multiplication and your implementation approach. When describing your project, consider the following questions as guidelines: What is matrix multiplication good for? How is it done numerically? Why would you want to parallelize it? Give an example where matrices containing random numbers appear.

The aim of your project is to multiply (somewhat) large random matrices and to compare the performance of sequential *vs*. parallel programming. The performance is measured in how much CPU time (wall clock time and not the CPU ticks) is needed to complete the task. To get started, familiarize yourself with matrix multiplication.

Follow the given tasks 1 to 5 below to complete this aim. If not noted otherwise, A is an n × m matrix and B is an m × p matrix in the following.

Task 1: Matrix multiplication

a) Write a sequential program to perform matrix-matrix multiplication X = A × B, where A is an n × m matrix and B is an m × p matrix, their matrix product A × B is an n × p matrix. Choose n=m=p=5. Print the output on the screen.
Example output:

```
This is a matrix multiplication (sequential code).
A=…
B=…
AB=…
```

b) Write a parallel program using openMP that does the same job.
Example output (e.g., assuming you have 8 cores on your machine available):

```
This is a matrix multiplication (parallel code). Number of
threads:8.
A=…
B=…
AB=…
```

Task 2: Matrix multiplication and summation

a) Write a sequential program to perform matrix multiplication X = A × B + C, where A is an n × m matrix and B is an m × p matrix. What dimensions has C? Choose n=5, m=4, p=3. Print the output on the screen.

b) Write a parallel program using openMP that does the same job.

Task 3: Random number generator

a) Fill a n × m matrix with uniformly distributed random numbers in the interval [0,1]. Choose n=5, m=4 and a seed region of 101. Print the output on the screen.

b) Write a parallel program using openMP that does the same job.

Task 4: Fill a larger matrix with random numbers

a) Fill a larger matrix with random numbers: choose n=m=1000. Do not print the output on the screen. Extra task (with bonus): Display a heat map of the matrix. What are heat maps good for?

b) Write a parallel program using openMP that does the same job.

Task 5: Integrating time measurements

With parallelized code, you should be faster. How much faster would you expect to be?

a) Write a sequential program that calculates the matrix-matrix product (A × B) + C for dimensions n=m=p=1000, including random numbers with a seed of 1234.

b) Write a parallel program using openMP that does the same job.

c) Write a program that compares both computation times (sequential code *vs*. parallel code in wall clock time). An example output should look like this:

```
This is a matrix multiplication of 1000 by 1000 matrices
CPU time (sequential code): 56.78 seconds
CPU time (parallel code with 5 threads): 12.34 seconds

The parallel program is x-times faster than the sequential
code.
```