Prof. Dr. Kai Diethelm
Faculty of Applied Natural Sciences and Humanities
University of Applied Sciences Würzburg-Schweinfurt

Summer Semester 2024

Parallel Programming
Assignment 4

Deadline: May 21, 2024, 23:59 CEST

General Notes:

- For the programming assignments, you may use either of the programming languages C, C++, or Fortran. On the FANG HPC system, the corresponding compilers can be invoked by the commands `gcc`, `g++` and `gfortran`, respectively.
- To submit your solutions, follow the instructions given in the course webpage in our Elearning system.

Computer Science Background: Quicksort is a very popular algorithm for sorting a list of objects of the same kind (numbers, character strings, etc.). It is being given an unsorted list of objects and returns a sorted list that contains the same objects. It is recursive in nature and works as follows:

- Given a list $a = (a_\ell, a_{\ell+1}, \ldots, a_r)$ of objects:
  - If $\ell \geq r$ then the list is empty or contains only one element. Therefore, it is already sorted, so nothing needs to be done. Return.
  - Otherwise:
    - Pick the last element $A = a_r$ of the list as the so-called *pivot element*.
    - Let $k = \ell$ and $j = r - 1$.
    - While $k < j$:
      - While $k < r$ and $a_k < A$: increment k by 1.
      - While $j > \ell$ and $a_j \geq A$: decrement j by 1.
      - If $a_k > a_j$ and $k < j$ then swap a_k and a_j
    - If $a_k > A$ then swap a_k and a_r.
    - *At this point of the algorithm, the list has been partially sorted in the sense that the pivot element has been moved to its correct position, all elements on the left of the pivot element are less than or equal to it (but not necessarily in the correct order yet) and all elements on the right of the pivot element are greater than or equal to the pivot element (and once again not necessarily in the correct order yet). Therefore, it remains to sort the two sub-lists on either side of the new position of the pivot element.*
    - Call the quicksort function with the list $(a_\ell, \ldots, a_{k-1})$.
    - Call the quicksort function with the list $(a_{k+1}, \ldots, a_r)$.

Questions:

1. (a) Write a function that implements a sequential version of the Quicksort algorithm as described above. Assume that the list to be sorted is a list of integer numbers.

   (b) Write a main program that
   - creates a list of one million random integers in the range between 0 and 100 000,
   - writes this unsorted list into a file,
   - sorts the list with your quicksort function, and
   - writes the sorted list into another file.

   (150 Points)

2. Parallelize your program from Question 1 with OpenMP. Provide detailed comments explaining your choice of the OpenMP constructs, including the handling of variables (shared vs. private). Provide a document that lists the run times of (a) the sequential program, and (b) the parallelized program when running with 2, 4, 8, and 16 threads.

   (250 Points)