Lab 3. OpenMP – parallel sorting and performance studies

Main goals of the assignment

- Learn about the differences between the efficiency and the improvement percentage using parallel version.
- Learn how to use a low number of processing elements in simulation of a large number of parallel tasks

The problem to solve

General overview

A parallel version of odd-even sorting that is efficient is requested.

Background

The odd-even sorting compares every 2 consecutive numbers in the array and swap them if first is greater than the second to get an ascending order array. It consists of 2 phases – the odd phase and even phase:

Odd phase: Every odd indexed element is compared with the next even indexed element.

Even phase: Every even indexed element is compared with the next odd indexed element.

See the bellow figure for an example:

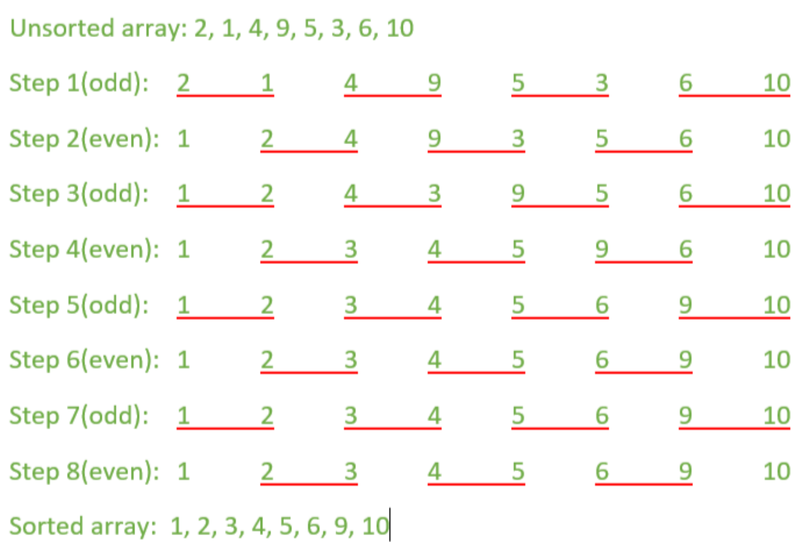

Figure 1: Example

How to parallelize

The maximum degree of parallelism is $n/2$. Usually the number of processing elements is lower than $n/2$. Then we need to try an almost equal distribution of the coomparisons-exchanges on our cores. How we can do?

To do

1. Write the sequential code to implement odd-even sorting.
2. Write the OpenMP versions:
   (a) group the $n/2$ comparisons-exchanges of one parallel step on the p threads;
   (b) create a thread for each comparison-exchange (do not take into account the fact that we have $p \ll n$ by using $n/2$ the number of threads)

   Introduce time records (hint: omp_get_wtime) before and after the part that is parallelized. Decide which version is optimal using a very large sequence to be sorted (milions order).
3. Record the times $T_p^{(n)}$ for the best variant in table like the folowing (with the maximum cores that you have, e.g. 8 or 16) – see the table

Table 1: Put inside the boxes the recorded times

$n\backslash p$	1	2	4	8
100000				
200000				
400000				

4. In order to compute the speedup we need to know the sorting time for the best sequential algorithm (quicksort!) Compute the speedups using

$$S_p^{(n)(odd-even)} = \frac{T_1^{(n)(quick-sort)}}{T_p^{(n)(odd-even)}}$$

   and record them in a similar table with the above one.
5. Draw conclusions related to:
   - is the parallel version of odd-even effcient?
   - dependence of the answer on the problem dimension n.