Parallel computing 12.1.2018
Exercise 1

Read the questions carefully, answer to all parts of each question. Draw a picture of each exercise. Submit the solutions to Moodle by 12.1. 09:00 (1 hour before exercises).

1. Design a manual message-passing sorting algorithm for a group of real persons and A4 papers. Each paper has a name according to which it is sorted. At the beginning, all papers are in a stack by the first person. At the end, the sorted papers should be returned to the first person. A person can perform local operations and give a stack of one or more papers for the neighboring person. The recipient needs not to be ready to accept the stack (post-pox messaging). No person can move from his/hers position.

   Assume that you have a) 10 persons, b) 100 persons located in a single row. Assume that you have A) 100 papers, B) 10 000 papers.

   Design the sorting algorithm (as fast as possible) to do the real-world sorting (in both a and b sitting arrangements and both A and B paper piles). Calculate the approximate time needed (in seconds) for all (Aa, Ab, Ba, Bb) cases.

   Hint: try to avoid situations when only one person is doing something useful.

2. Modify the previous case b) algorithm so that instead of a single row, the 100 persons are located in a 10 by 10 matrix, and are able to pass the papers in 4 directions. Take advantage of the reduced diameter of the human network. Calculate the time again for both A and B cases.

3. Design an algorithm for 10 000 papers and 1000 persons. Calculate again the time required. Persons can move freely and you can (have to) define how they are located. Notice, however, the physical space requirements for 1000 persons. Hint: try to avoid situations when only few people are doing something elaborate.

The following tasks are best solved by spreadsheet formulas. Bring your sheet to the exercise session..

4. Let us consider an computational problem that can be solved sequentially in time n^3 (e.g., matrix operations for $n \times n$ matrices) and unit time is one ns (10^{-9}s). Thus, e.g., for $n=1000$, the computation would take $1000^3 \times 1\text{ns} = 1\text{s}$. Let us first assume that our parallel algorithm is fully work efficient, i.e., parallel computation will take n^3/p (p = the number of processors/cores). How much larger inputs we can handle in 1) 1 second, 2) 1 year, if we have a) 8, b) 1000, c) 1 million processors available? How many processors we would need if we would like to solve the problem for $n = 10$ million in a year? Do a quick www-search and estimate how much such machine would cost? Make a coarse estimate of euro/core.

5. Let us continue the previous task, but now our algorithm is inefficient by factor of
   a) 2, b) loglog$n$, c) log$n$, d) $\sqrt{n}$, e) n, f) $\sqrt{p}$.
   (logarithms are base 2, $\log_2 1000 = 10$). E.g., for $n=1000$, $p=100$, inefficiency = log$n$, the computation would take $1000^3 \times \log 1000 \times 1\text{ns}/100 = 0.1\text{s}$. Now, how many processors in each case (a-f) we would need to solve the problem in a year for $n = 10$ million.