Parallel Computing
Exercise session 2 (12/11/2012)

Ex. (1) Amdahl's law.
Consider a 'master-slave' algorithm, in which the master processor first distributes work to the slaves, and at the end collects and processes the results produced by the slaves. Assume that there is no other overhead (communication, idling, ...). Assume that the master's task takes a factor s of the total execution time. Calculate the speedup and the parallel efficiency for $p = 20$ and $p = 1000$ if:

(a) $s = 0.1$

(b) $s = 0.01$

(c) $s = 0.001$

(d) Suppose that for the same algorithm you measure a speedup of $S = 3.8$ on 4 processors. Which speedup do you expect on 64 and on 128 processors?

Ex. (2) Parallel dot product.
Consider the two parallel dot product algorithms studied in the previous exercise session. Assume that n is multiple of p.

(a) Determine the overhead function and parallel efficiency.

(b) Assume that g, l and n are constant. How do the algorithms scale with the number of processors?

(c) Assume that g is constant and l is linear with p. How do the algorithms scale now?

(d) On Cray T3E computer, for $p = 64$ we have $g = 78$ and $l = 1825$. Calculate the parallel execution time. What can you say about the parallel efficiency?

Ex. (3) Grid-based stencil operations.
Consider 2D and 3D rectangular grids, with M grid points and a stencil operation (2D: five point stencil; 3D: seven point stencil). Consider p processors and 1D,2D (and 3D) partitionings. Derive formulas for the parallel runtime (T_p) and the parallel efficiency (E). Discuss the dependence on M, p and $n = M/p$ (number of points per processor).

Ex. (4) Parallel odd-even transposition sorting algorithm (variant of bubble sort).
See slides of the lecture of 9/11/2012 for the description of the algorithm. Assume that we sort an array of n elements on p processors (or 'cores'). What is the parallel run time in the BSP model? Derive an expression for the parallel efficiency E that clearly shows how E depends on p, n and the problem size. How should the problems size grow with the number of processors p to have iso-efficient behavior?

Ex. (5) The sieve of Eratosthenes:
The sieve of Eratosthenes is an algorithm for finding the prime numbers contained in the interval $[2, n]$. Try to understand the given sequential version of the algorithm and address the following tasks:

(a) Consider cyclic and block partitionings of the data for parallelization purposes. Modify the given algorithm and obtain a new version for each of these strategies.

(b) Discuss the advantages and disadvantages of using cyclic and block distributions. Which one is the best? (consider that n is large enough so that $\lceil n/p \rceil > \sqrt{n}$).

(c) Consider the best partitioning strategy. Assume that the total parallel overhead is zero. Why can't we achieve an ideal speedup $S = p$?