Exercise Sheet 1

Think Parallel

Lecture *Parallel Computing Systems*, Winter semester 2025/2026

Prof. Dr.-Ing. Mladen Berekovic
A discussion forum for the exercise can be found at: `moodle.uni-luebeck.de`.

Submission Guidelines

Please follow the instructions in "Submission Guidelines" document published within this course.

Think Parallel (20 pt.)

Given the following array x composed of 8 distinct elements:

4 9 2 7 1 6 3 8

1. Design both a sequential and a parallel algorithm for computing the prefix sum.
   Reference : Blelloch, Guy E. *Prefix Sums and Their Applications.* School of Computer Science, Carnegie Mellon University.
2. Apply the reduce operation to the input array and report the outcome.
3. Determine the output of the down-sweep phase.
4. Evaluate the time complexity, operation count, and the number of required CPUs for both strategies.

Parallel Algorithms (20 pt.)

In this exercise, we will be dealing with the scalar product (dot product) of two vectors.
Given: A and B, two vectors with 200 elements each.

1. Illustrate in a scheme or a simple algorithm the sequential scalar product of A and B.
2. Illustrate in a scheme or an algorithm of a parallel version with **8 processors** to solve the task.
3. Determine the time steps of both sequential and parallel processing.
4. Calculate the speed-up S_8 and the efficiency E_8.
5. Generalize S_p and E_p as functions of the vectors length **n** and the number of used processors **p**.

PRAM - Parallel Random Access Machine

Matrix Multiplication (20 pt.)

Define A, B, and C as matrices of dimensions NxN, meaning the majority of their elements are nonzero.
A matrix-matrix multiplication is illustrated as follows:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \times \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix}$$

$$= \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$$

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} * b_{k,j}$$

1. Modify the algorithm to adapt it to EREW PRAM Model. Calculate the time complexity and the number of used processors.
2. Design an algorithm that works with **O** (n^2) processors. Calculate the time complexity, the speed-up and the efficiency. Compare and discuss the results.

Distributed Search (20 pt.)

Consider a string with length of **n** unique characters.
To search for a character **c**, a sequential approach will require up to $\mathbf{O}(n)$ time in the worst case.

1. Given **p** processors, derive an efficient parallel algorithm for this task using the following models:
   - **EREW-PRAM**
   - **CREW-PRAM**
   - **CRCW-PRAM**
2. Determine the time complexity for each of the previous algorithms.

Programming Assignment (20 pt.)

Implement the “Distributed Search” problem using your preferred programming language. Then, record a video demonstrating the execution and explaining your solution.
Implementation Requirements:

1. Provide both sequential and parallel solutions using the same input data.
2. Present the speedup visually (e.g., plots, charts,...).
3. The recording should clearly display the source code, output results, and your explanation.