# Project — application of data concurrency tools

## Grading

- Report — max 5 points
- Program implementation — max 2.5 points
- Answers to defence questions — max 2.5 balo

## Structure of the report

1. **Title page and description**.
2. **Task analysis and solution** - briefly describe the problem being solved and how you have solved it, how data concurrency tools were applied.
3. **Testing and instructions** - show that the program outputs **correct** result and provide an instruction how to launch your program.
4. **Performance analysis**:
   (a) Prepare at least 8 datasets of different size (it is recommended that the smallest dataset is processed faster than in 10 ms by a single thread, the largest - longer than 2 minutes).
   (b) Find an optimal amount of threads or processes for each dataset, show diagrams that reveal how execution time changes depending on dataset size and thread or process count. Measure execution time several times and provide averages.
   (c) Explain your results: comment your diagrams, describe why do you think you got exactly your results, not something different.
5. **Conclusions and literature** - what did you learn, what did you notice that was expected or unexpected, comment how well your tools fit for the selected problem.

**Tools** Goal — try tools for data concurrency that control threads automatically but allows setting thread count. You may choose the tools of your preference (must be approved by your professor) or one the following:

- C++ with OpenMP `parallel for`
- C++ with MPI `Scatter` / `Gather`
- C++ with CUDA
- Haskell `rpar` and `parListChunk` or another strategy
- Python `multiprocessing.Pool`
- Ruby with parallel gem
- Java / Kotlin `parallel stream`
- C# `parallel LINQ`
- Scala `parallel collections`
- F# `PSeq`
- Rust `rayon`

## Submission and defense

- Your defense date is provided on Moodle, you have to defend at your assigned week.
- If you want to use a tool that is not on a recommended list, they must be approved by professor.
- Submit you reports in PDF.
- Upload your report, program and data files (1 dataset) to Moodle before project defense.
- If you failed the defense, repeated defense is on week 16.

## Task

- If you have numeric methods course, it is recommended to parallelize your second homework optimization task.
- You may choose your own task, but is has to be approved by your professor in advance. The task must be a data concurrency task, i.e., the same operation is applied to all items in your dataset and some result is computed.

## Requirements for the program

- It must be possible to change thread or process count independently from your selected dataset;
- It must be quick to change the used dataset and thread count during your defense;
- Goal of the project is to get as good speedup as possible while making your code as simple as possible;
- When you perform your performance analysis, do not run your program in debug mode, e. g., if you work with Visual Studio and C++, change your Debug configuration to Release — compiler will turn on all optimizations and your program may run faster.