# Parallel & Distributed Computer Systems 2023

# Homework 2

# Distributed k-select

### Deadline: January 13, 2024

### Introduction

As we discussed in class, the MPI homework is the distributed k-select using MPI. First, we write a sequential function to do partition in place, as in this MATLAB.

```
function [p,q,A] = partitioninplace(A,v)
% function [p,q,A] = partitioninplace(A,v)
% reorder array A(1:n) and find p and q so that after the call,
% A(1:p-1) < v, A(p:q-1) == v and A(q:n) > v
% for any value v that may or may not be in A(1:n)
```

Use the above function to implement `kselect(A,k)` that finds the kth smallest value of array `A`. Integer `k` is in the range `1:n` and `A` is an array of length `n`. The function should return the value `A(k)` of a sorted `A`, without sorting `A`.

### Parallel implementation (C/C++)

Once your model code in MATLAB/Python/Julia/R (whatever allows you to quickly prototype and confirm correctness) works, implement it in C/C++.

Suppose we had a very large file holding the array `A(1:n)` (that exceeds the memory of a computer). Extend your sequential code with MPI into a distributed program for `kselect(A,k)`, where `p` is the number of processes participating in the computation. Each process `1:p` works with its own data, but it coordinates globally for every decision how to proceed with the search. It is not worth to redistribute the data unless `n/p` fits in a single memory.

Extra credit if you use multiple threads in each process to parallelize `partition` and take advantage of the shared memory of a multicore processor. A multi-threaded solution will need to be using a ping-pong buffer as I think it will be too hard to do it in place and in parallel and faster :smile:. Feel free to use any threading compiler or library but make sure it is compatible with the MPI library you are using.

### Alternative Parallel implementation in Julia

If you elect to implement the whole project in Julia and not C/C++, the multithreading won't be optional. Let's discuss what packages to use. If you'd like to use some other language, please coordinate with me first.

## What to submit

- A 4 page report in PDF format. Report execution times and speedup of your implementations with respect to the number of items $n$, and the number of processes $p$. Please identify how many different host servers are used. MPI behaves differently if the processes are on the same host or on different hosts.
- Upload the source code on GitHub, BitBucket, Dropbox, Google Drive, etc. and add a link in your report.
- Argue about the validity and effectiveness of your code. You can also include any other information you consider important for the evaluation of your work.
- Cite any external sources you may have used.