# Slurm 101

How to use the HPC-infrastructure at CIMH

Moritz Werr

Research IT

May 19, 2025

## Outline

## Introduction

- Modern science uses computers to process data
- Datasets get bigger and bigger
- More processing power is required to process data in a timely manner

$\rightarrow$ Introduction of HPC-hardware at CIMH for fast data processing

**HPC-Cluster for parallel computing**

The Research IT administers a Cluster for **High Perfomance Computing** (HPC). Each machine inside the cluster has a lot of CPU-Cores and RAM, some even GPUs, for parallel data processing.

## Introduction

### Outsource your number crunching

Our HPC-cluster is configured for batch processing. Submit your data processing jobs to the HPC-queue and it run on the cluster. Most HPC-machines have more hardware resources than individual analysis machines.

### HPC introduces some complexity

An HPC-Cluster is a distributed system, ie. it requires more knowledge than running data processing locally. Additionally you are running programs in parallel, which can speed up workloads substantially, but also waste a lot of computational resouces.

### HPC is a shared medium

The HPC-Cluster is shared between all users in the institute. It is common courtesy to use up as little resouces as necessary. Test your workloads beforehand and don't request more resources than you need.

## Parallel Computing

- Computer Programs contain multiple steps
- Some steps can only be run **serially**, some can be run in **parallel**
- Serial parts are hard to speed up
- Parallel parts can be speed up fairly easy

**Careful!**

Trying to parallelize serial steps can lead to a decrease in performance, crashes and wrong results!

## Parallel Computing

- Parallelization requires knowledge of the workload and your program
- Which parts can be parallelized depends on your data and program code
- Parallelizing wrong parts of your program can cause **Race Conditions** and **Deadlocks**

**Don't waste resources**

Simply adding more computational resources won't increase calculation speed.

## Parallel Computing

Each Program has steps which can be run in parallel and steps which can only be run serially.

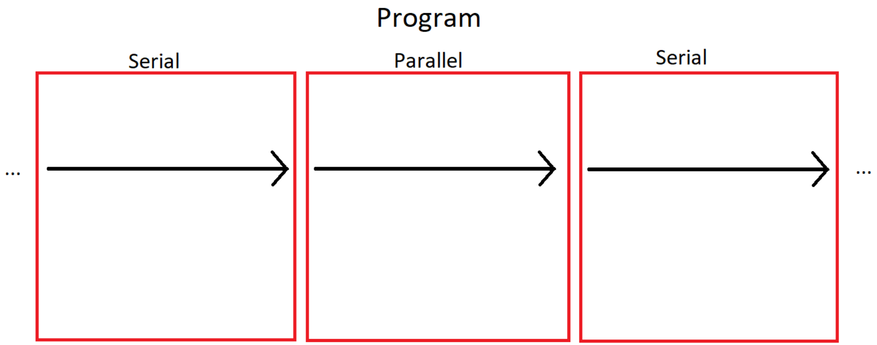

Figure: Programs have serial and parallel parts.

# Parallel Computing

Adding more resources at the wrong step will have no positive impact on overall performance. In fact it might cause problems.

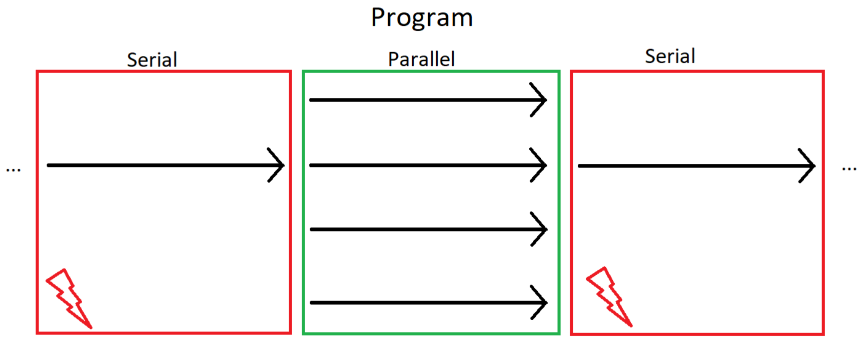

Figure: Parallel parts benefit from more computing resources, serial parts don't.

## Parallel Computing

There are multiple ways to parallelize your program:

1. Vectorization
2. **Multi-Processing**
3. **Multi-Threading**
4. Multi-Streaming with GPUs

**Right tool for the job**

Choosing the right parallelization for your program is important. In these slides I will focus mainly on Multi-Processing and Multi-Threading.

## Parallel Computing

### Split up your workload

Parallel Computing requires you to split up a **larger problem** into **smaller sub-problems**, which can be calculated independently.

### Vectors, Matrices, Tensors

Vectors, Matrices and Tensors operations can be parallelized very easily as each element can calculated independently from the others.

### Setup your data

Splitting up a large datasets into smaller independent datasets is key for parallel execution. Each dataset can be computed on different Nodes inside the cluster.

## Parallel Computing

Not every dataset can be split up in the same way. Each subset must be independently computable.

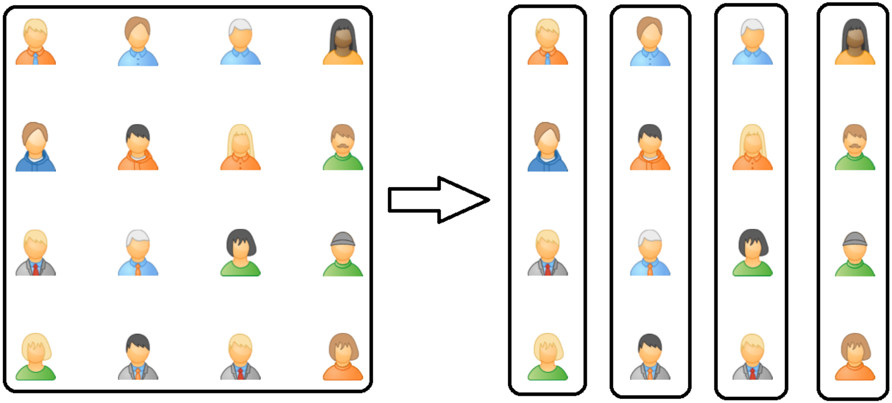

Figure: Splitting up a dataset of multiple subjects into smaller subsets.

## Parallel Computing

- After splitting up your problem into smaller subsets, you can work on them in parallel
- You can use multiple threads with a shared memory space to work with multiple cores on a single subset
- You can use multiple processes with their own separate memory space to work with multiple cores on different subsets

## Parallel Computing

Multi-Threading

- Memory is shared between multiple CPU-cores
- All CPU-cores work on the same variables
- Safety mechanisms for Race Conditions and Deadlocks can slow down computation
- Serial parts usually results in all but one CPU-cores are idling

Multi-Processing

- Multiple CPU-Cores work simultaneously in their own memory space
- Each CPU-Core has his own set of variables
- No slow down due to safety mechanisms or parallel parts
- No Perfomance gain for individual runtime

## Parallel Computing

Using multiple threads can speed up the runtime of individual subsets. Multiple CPU-cores can work on the same data due to a shared memory space. During serial parts of your program, additional CPU-cores are idling.

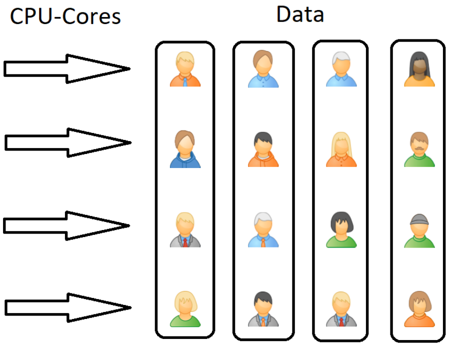

Figure: Multiple threads working together on the same subset.

## Parallel Computing

Using multiple processes doesn't speed up the runtime of individual subsets. However each process can run completely serially. No CPU-cores are idling during serial parts.

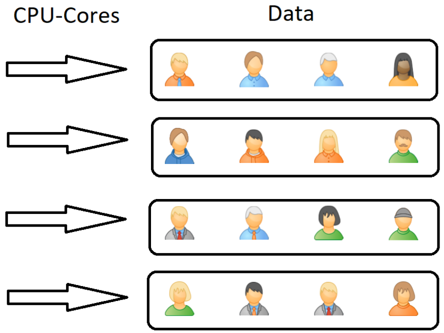

Figure: Multiple processes work independently on different subsets.

## Parallel Computing

When to use multi-threading:

- Speeding up individual datasets, which can't be split up further/splitting up further is very hard.
- Individual runtime is very important.
- Your computation has very small serial parts/big parallel parts.

When to use multi-processing:

- Datasets can be split up into very small subsets
- Individual runtime is not important, overall runtime needs to be quick
- Your program doesn't benefit from a shared memory space

**Try out what works best**

Parallel computing is very complex and requires a lot of testing. Try out different levels of parallezitaion and even combine both methods. Test your job-scripts on smaller subsets and look how they impact runtime.

## Our HPC-Cluster

- 24 virtual nodes and many other hardware nodes (old hardware getting reused)
- Access via your RDS-Machines
- Local storage and FIstorage+Home shared
- Resources are CPU, RAM and GPU

## Basic Tools

- sinfo/squeue Status
- salloc/srun/sbatch Nutzung
- sacct Analyse

## Requesting Resources

- Default Values
- Absolute/relative
- CPU/RAM/GPU

## Interactive Mode

- Interactive mode for testing
- srun to run commands on cluster
- salloc for longer session

## Batch Processing

- Kinit -¿ copy input -¿ process -¿ Kinit -¿ mv output
- Request ressources in script
- Mail

## Kerberos

- Kerberos on RDS (Login -¿ no manual kinit needed)
- Create Keytab
- Kinit in script

## Examples

- Normal
- Array

Thank You!