All things computing with parallelization and CRC

March 17, 2026

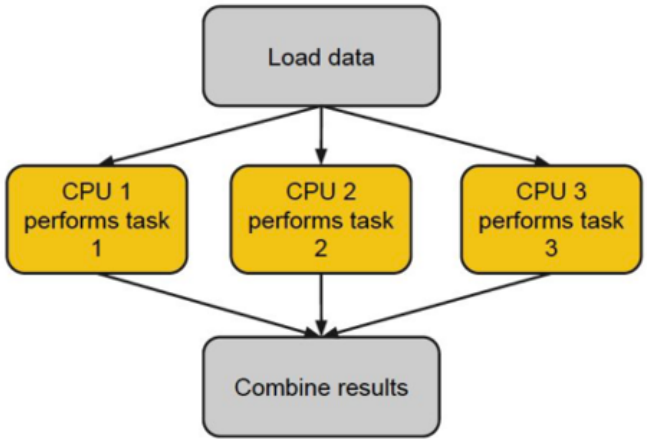
Parallel Roadmap

1. **Introduction to Parallelization**
2. **How to Parallelize Locally?**
3. **How to Parallelize on CRC (remotely)?**

Introduction to Parallelization

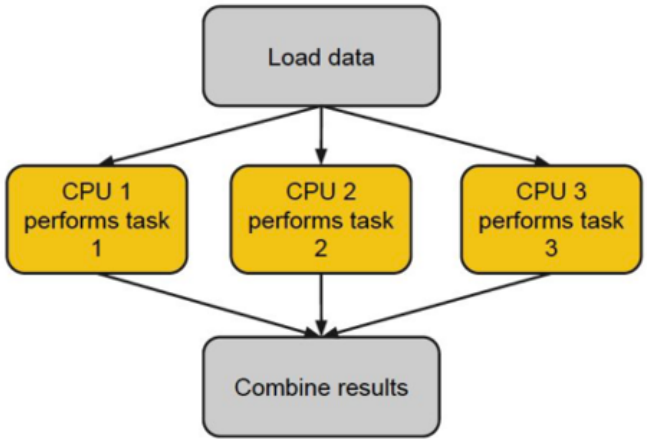
Parallelizing Code

- **Parallelization**: performing computational tasks simultaneously (as opposed to sequentially)



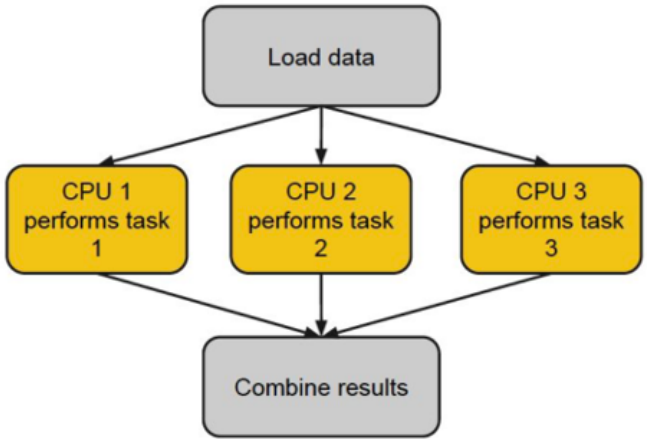
Parallelizing Code

- **Parallelization**: performing computational tasks simultaneously (as opposed to sequentially)
- *Key requirement:* parallel tasks *cannot* talk to or depend on one another



Parallelizing Code

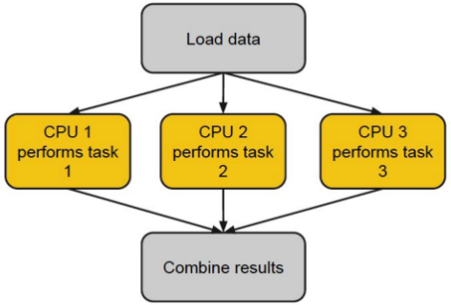
- **Parallelization**: performing computational tasks simultaneously (as opposed to sequentially)
- *Key requirement:* parallel tasks *cannot* talk to or depend on one another
- We will focus on **embarrassingly parallel** tasks



Example of an embarrassingly parallel task

- Imagine we have a for loop where **each iteration of the for loop does not depend on any other iteration** of the for loop, e.g.,

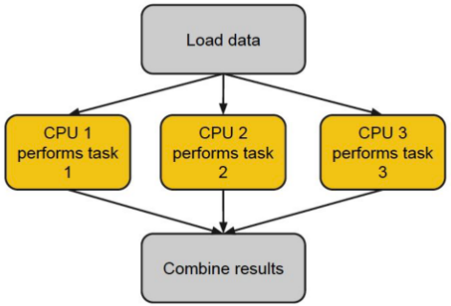
> for each b = 1:B:
>
> Take a subsample of the data matrix X
>
> Do something with the subsample (e.g., fit model)
>
> end



Example of an embarrassingly parallel task

- Imagine we have a for loop where **each iteration of the for loop does not depend on any other iteration** of the for loop, e.g.,

> for each b = 1:B:
>
> Take a subsample of the data matrix X
>
> Do something with the subsample (e.g., fit model)
>
> end



- Rather than computing this for loop sequentially, we can run each iteration “in parallel” (i.e., simultaneously) using different cores

How to Parallelize Locally?

R Parallelization Tools/Resources

- **future**: https://future.futureverse.org/articles/future-1-overview.html
- **furrr**: https://furrr.futureverse.org/
  - Offers drop-in replacements for parallelizing purrr::map() functions
- **future.apply**: https://future.apply.futureverse.org/
  - Offers drop-in replacements for parallelizing base R apply() functions
- **doFuture**: https://dofuture.futureverse.org/
  - The “futureverse” version of the doParallel R package
- Stay up-to-date with the latest developments in the futureverse: https://www.futureverse.org/blog.html

Python Parallelization Tools/Resources

- **joblib:** https://joblib.readthedocs.io/en/stable/
- **multiprocessing:** https://docs.python.org/3/library/multiprocessing.html
- **Dask:** https://www.dask.org/
  - Great for handling large datasets and scaling operations from a single machine to a cluster
- **Ray:** https://www.ray.io/
  - Useful for building distributed applications and handling complex parallel tasks across multiple machines
- A great introductory tutorial covering joblib, multiprocessing, and Dask from Thomas Langford (Yale): https://docs.ycrc.yale.edu/parallel_python/#/

Parallelization example **locally**

Git pull from our course `dsip-s26` repository

All relevant code can be found in `course_materials/parallelization/` directory

For R users:

1. Open R project: `parallelization/parallelization.Rproj`
2. In R console: `renv::restore()`
3. Open **`notebooks/parallel_example_R.qmd`**

For Python users:

1. In terminal: `cd path/to/course_materials/parallelization`
2. In terminal: `conda-lock install --name dsip_parallel`
3. Open **`notebooks/parallel_example_python.qmd`** and use `dsip_parallel` conda environment

Rendered R+Python output in **`notebooks/parallel_example.html`**

How to Parallelize on CRC (remotely)?

Getting started on the CRC

Go to `parallelization/notebooks/crc.html`

How to submit a job on the CRC

1. **Clone GitHub repositories and copy over all necessary files to CRC**
2. **Write a script that you want to run**
3. **Make sure that the necessary dependencies are installed**
4. **Write a job submission script**
5. **Submit the job**
6. **Monitor the job's progress**

Interactive Jobs

There are many possible ways to work interactively on the CRC:

- Via **command line**
- Using **Rstudio**
- Using **JupyterLab**
- Using **VSCode**

Set up for each tool is detailed in crc.html