Lecture 18 – Parallel processing in R

Learning Objectives:

1. **Become proficient in the use of the R language**

    1.9 Learn how to parallelize R code.

3. **Learn the basic principles of software design.**

    3.8 Learn the basic principles of parallel computing.

Midterm survey is on Canvas!

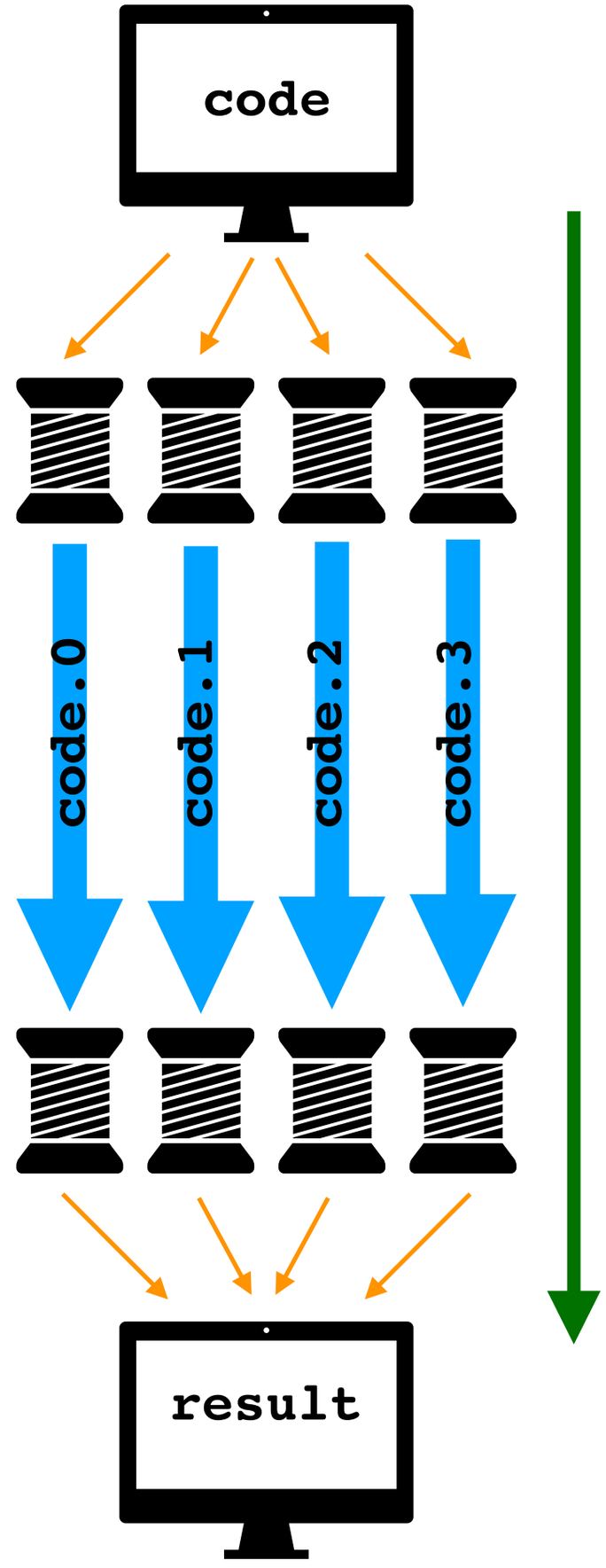
Basics of Parallel Computing

- Most modern computers have multiple CPU units (cores) that will process data independently.

- There are helpful packages that will let you take advantage of this parallel processing in R, speeding up calculations but not doing the calculations faster but breaking them up so that each core calculates part of the job.

- In parallel computing, you decide how to break up the code into smaller pieces that can run independently.
  - The kernal then assigns the code to each core and these cores then runs the code.
  - When each piece is done, the kernal must deliver the results back to R and stitch them together.

- Parallel computing improves **elapsed time** (or “wall clock” time, how much time passes for you), by increasing **user time** (the sum computing time of the cores) and **system time** (the time spent splitting up and stitching together the results).



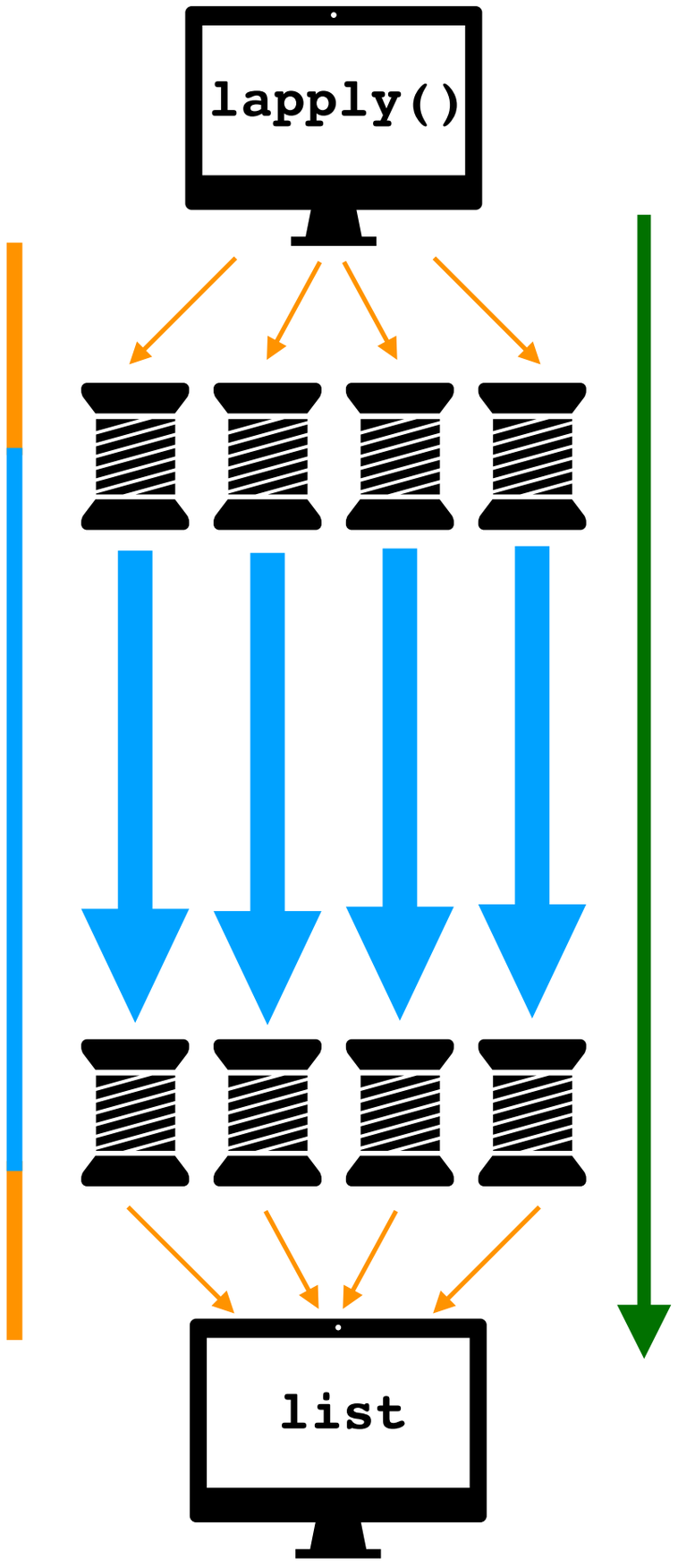
Two extremes: “embarrassingly parallel” calculation

- If a calculation is broken up and each piece runs independently, it can be *much faster* in parallel, these are often called “embarrassingly parallel.”

- Each calculation is run completely independently, so elapsed time decreases *very quickly* with an increasing number of cores.

- system time is negligible, `lapply()` returns a list of results so it doesn’t even need to combine anything!

- `lapply()` is an example. Run these with the sample code that uses the `parallel` library.



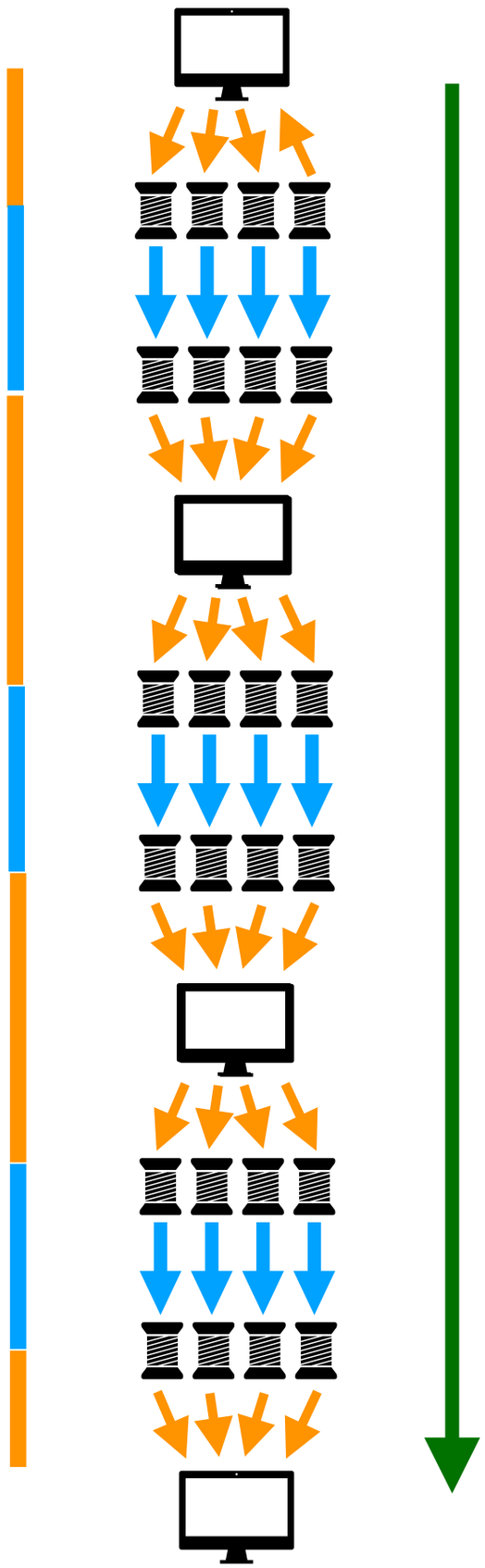
Two extremes: calculation that scales poorly

- Breaking up and stitching together takes *time*. This results in some calculations varying in how much elapsed time they pick up for how many cores you use.

- As each process is broken up and stitched together, it uses system time (which is always serial). More cores means more time breaking apart and then stitching together, so after awhile, you're using more system time than you're saving by running calculations in parallel!

- This is called "scaling" and depends on the function and how you break it up. High-performance computing resources will often request **performance scaling** as a part of proposals, so that they know you are using an appropriate number of cores for your code.

- An example of code that doesn't scale well is a calculation in a for loop in which every step depends on the step before.



What affects parallel performance?

- The calculation itself

- How you break up code to run

- The speed of individual processors

- How code is accessing memory

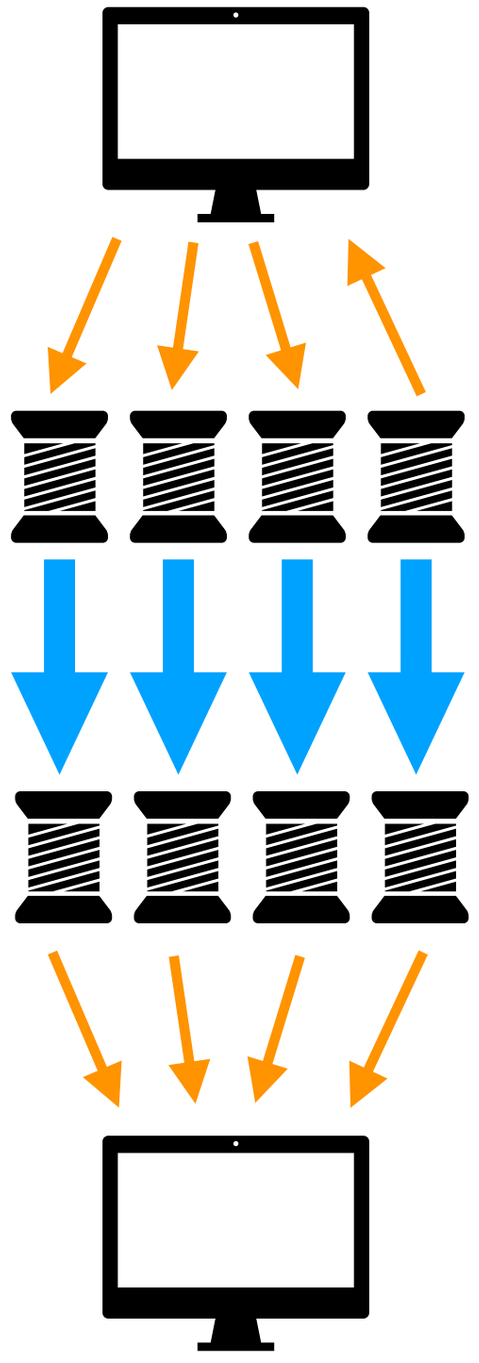
- The connections between hardware in clusters

- Lots more we won’t cover here!

Parallel Computing in R

- Example R code in folder with `parallel` package, part of base R. (Note differences between Windows and Mac/Linux)

- Example R code in folder with `foreach` package, parallelizing code based on for loops.
  - `foreach` is useful because it is platform independent!



Parallel Computing Exercise

- Use either the parallel or foreach package in order to parallelize the AdvDiff simulation.

- Run the code with **100,000 dots** for a total time of **10 seconds** with a **1e-3 second** time step.

- Compare the amount of time it takes in serial versus parallel.

Questions to consider:

- How will you break this code up to run?

- What aspects of the code will you need to change in order to return the same output?

- Which package will work best with the least amount of code change?

Additional Resources

http://adv-r.had.co.nz/Profiling.html#parallelise –
Optimization: parallelise, in *Advanced R*

https://cran.r-project.org/web/packages/foreach/vignettes/foreach.html –
foreach package vignette

https://citeseerx.ist.psu.edu/viewdoc/download?
doi=10.1.1.467.9918&rep=rep1&type=pdf –
***Parallel Computing for Data Science*, by Norman Matloff**