Parallelization in R

Irina Degtiar[1] & Anna Zink[2]

[1] Department of Biostatistics, Harvard T. H. Chan School of Public Health
[2] Department of Health Care Policy, Harvard Medical School

Help! My code’s too slow :(

- How come?
  - Profiling & benchmarking
- How do I fix it?
  - Strategies
- How do I parallelize in R?
  - Hardware & terminology
  - Data vs. task parallelization
  - Forking vs. sockets
  - Packages in R
  - Example

Acknowledgements

- Christine Choirat: BST 262

Help! My code's too slow :(

- How come?
  - Profiling & benchmarking
- How do I fix it?
  - Strategies
- How do I parallelize in R?
  - Hardware & terminology
  - Data vs. task parallelization
  - Forking vs. sockets
  - Packages in R
  - Example

Profiling & Benchmarking

- **Profiling**: finding bottlenecks
  - Time
  - Memory
- **Benchmarking**: removing bottlenecks
  - Can optimize for total time, CPU time, memory usage, I/O, etc.
  - Tradeoff between runtime and researcher time
  - Run code multiple times to compare distributions

Profiling & Benchmarking: R

- **Profiling**
  - Base R: `system.time()`, `Rprof()`, `Rprofmem()`
  - R packages: `profvis`, `profr`, `lineprof`, `proftools`, `GUIProfiler`, `aprof`
- **Benchmarking**
  - R packages: `microbenchmark`, `rbenchmark`

Help! My code's too slow :(

- How come?
  - Profiling & benchmarking
- How do I fix it?
  - Strategies
- How do I parallelize in R?
  - Hardware & terminology
  - Data vs. task parallelization
  - Forking vs. sockets
  - Packages in R
  - Example

Strategies to optimize memory

- More efficient code
  - Vectorize
    - Avoid series of scalars
      - E.g., `runif(n)` rather than `runif(1)` within loop
    - Avoid concatenating within loop
      - E.g., `x[i] <- new_value` rather than `x <- c(x,new_value)`
        - Note: subscripting often will create a copy but will be faster than growing the object
  - Avoid copies
    - R creates (temp) copies when mutating/modifying object
  - Only use what you need: subset, `rm()` objects you don't need, use iterators
- Better data storage
  - Matrices rather than `data.frame`
  - `Matrix:` sparse matrices
  - `data.table`
  - `bigmemory, ffbase, diskframe, bigstatsr`

Strategies to optimize time

- More efficient code
- Faster languages/packages
- Parallelization

Strategies to optimize time

- More efficient code
  - Vectorize
    - Use vectors not scalars
    - Avoid for loops, use `apply` family or `purrr::map()`
    - Use vectorized functions (e.g., `rowSums()`, `colMeans()`, matrix algebra)
    - Vectorize your own functions: `Vectorize`
  - Be specific
    - Use functions specific to your data type
      - E.g., `vapply` instead of `sapply`
    - Specify arguments to function
      - E.g., specify column type for `read.csv()` or levels for `factor()`
  - Pre-compile your compute functions: `compiler::cmpfun()`
- Faster languages/packages
- Parallelization

Strategies to optimize time

- More efficient code
- Faster languages/packages
    - Faster all-around: Rcpp
    - Faster I/O
        - `fst`: data.tables/data.frames
        - `feather`: data.tables/data.frames (Python, Spark compatibility)
        - qs: all objects
        - Trim down models: `strip`
    - Look for other existing packages, especially for your specific task
- Parallelization

Strategies to optimize time

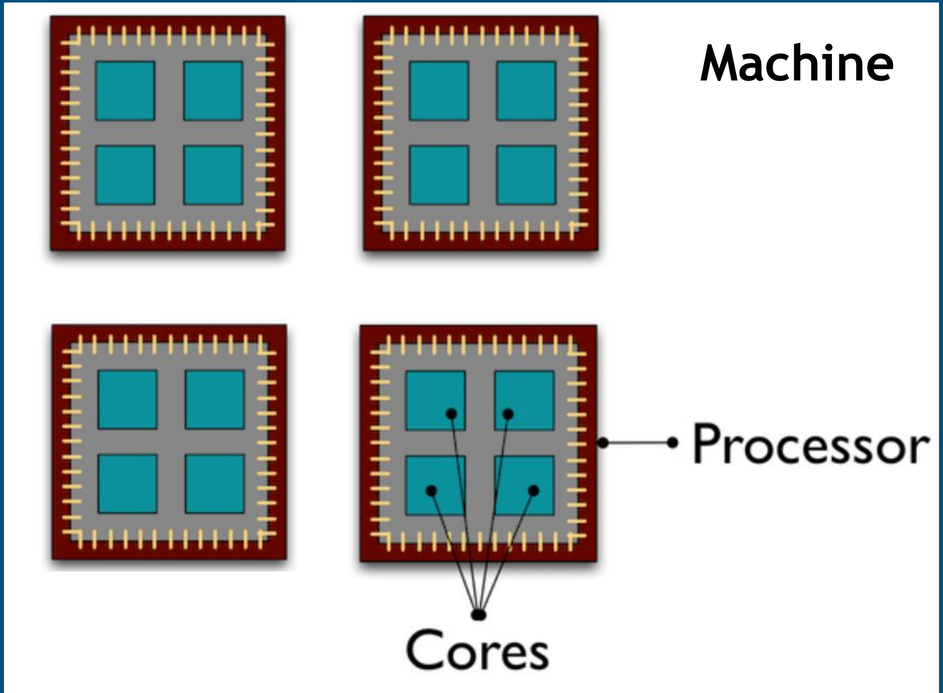
- More efficient code
- Faster languages/packages
- Parallelization
    - Words of warning
        - Remember overhead: I/O + thread creation
        - Servers often have worse machines than your laptop
        - Can make memory issues worse

Help! My code's too slow :(

- How come?
  - Profiling & benchmarking
- How do I fix it?
  - Strategies
- How do I parallelize in R?
  - Hardware & terminology
  - Data vs. task parallelization
  - Forking vs. sockets
  - Packages in R
  - Example

Hardware & terminology

- Machine, node, cluster, server
  - Machine/node = single physical computer
  - Cluster = interconnected computers
  - Server: lets computers work together as cluster
- Core, processor, CPU, socket
  - Core = basic computational unit
  - Processor/CPU/socket: can have multiple cores
- Parallel vs. distributed computing
  - Same machine = parallel computing
  - Different machines (cluster) = distributed computing
- Master and workers
  - Master/parent: orchestrates processes across workers
  - Worker/child/slave: each of the cores, does the work



Data parallelization

- Divide data into partitions and perform same task on every partition
- Use cases
  - Data can’t fit in memory
  - Data lives in different places (e.g., Sentinel)
  - Speed up computation when results combinable (e.g., bootstrap, sum or means)
- R
  - `tidyverse` split-apply-combine: `data %>% group_by() %>% summarize()`
  - `MapReduce`
  - `Divide and conquer`

Task parallelization

- Divide independent tasks across processors and perform different task on every data copy
- Use cases
    - Ensemble of algorithms (`SuperLearner`, random forests)
    - Comparison of statistical models

Forking vs. sockets in R

- Forking
  - Every worker inherits properties of existing R session (data, packages, etc.)
  - Data isn't copied until it's modified
  - Usually faster than sockets
  - Doesn't work on Windows (only Linux/Unix/OSX)
- Sockets
  - Every worker gets new version of R
  - Need to explicitly set up cluster, including loading packages: `cl <- makeCluster(4)`
  - Workers can communicate via network sockets
  - Works on Windows too

Popular Packages in R for Parallelization

- parallel
    - mclapply vs parLapply
- foreach
- doParallel

Beyond your laptop

- Software for parallelization
  - H2O
  - Spark
- Parallelization in the cloud
  - AWS, Google Compute Engine, Azure
- Parallelization on clusters
  - SLRM

Practicing Parallel Programming

Example code to get you started!

https://github.com/zinka88/hpds-parallel

Further resources

- Profiling
  - http://adv-r.had.co.nz/Profiling.html
- Fun read of R frustrations & solutions
  - http://www.burns-stat.com/pages/Tutor/R_inferno.pdf
- Parallelization
  - https://cran.r-project.org/web/views/HighPerformanceComputing.html?fbclid=IwAR3_RJ_09QcZTUz0MvCXRYOc3YZ_oTj1xcV9p_Gh4U0Iyjjd4u4QJeymfhE
  - http://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/notes/parallel.html
  - https://www.r-bloggers.com/2018/09/simple-parallel-processing-in-r/
  - https://bookdown.org/rdpeng/rprogdatascience/parallel-computation.html