Going Fast in R

Using Parallelization, Rcpp, and Julia

Jordan Love

4/18/2020

What do?

- Parallelization
  - Principles
  - Implementation (using `doParallel`)
- Alternative Languages in R
  - C++ using `Rcpp`
  - Julia using `JuliaCall`
- Measuring Improvement

Why for?

- Speed
- Access to language specific libraries outside of R
- Foundations for GPU Computing

Principles of Parallelization

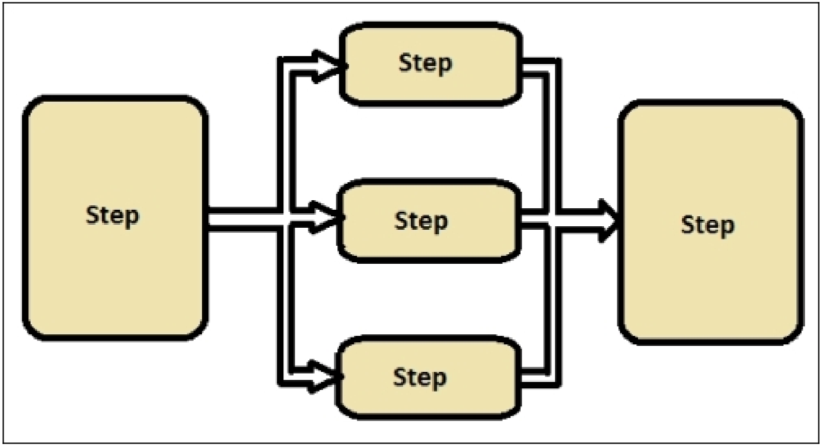

Figure 1: A Visual Example of Parallel Processing

Tasks of the type shown in this example are often called "Embarassingly Parallel"

Common Applications of Parallel Computing

- Image Problems (Neural Networks, Image Manipulation)
- Parallel Algorithms
- Large Datasets (Distributed Computing)
- Almost any problem that needs to go fast(er) and has many repetitive subtasks

Great, how does it work?

```r
simulate_covid19 <- function(county){
  ...
}

for(county in county_list){
  simulate_covid19(county)
}
```

Great, how does it work?

```r
library(doParallel)
library(foreach)

simulate_covid19 <- function(county){
  ...
}

num_cores <- detectCores()
cl <- makeCluster(num_cores)

foreach(county in county_list) %do% {
  simulate_covid19(county)
}

stopCluster(cl)
```

Short Tutorial: Parallelization

Let's show it in action.

Parallelization for Python Users

- The library `multiprocessing` provides basic parallel processing abilities
- Not as easy-to-use due to Python being a more general purpose language
- Good examples & introduction found at `https://docs.python.org/3.8/library/multiprocessing.htm`

Using Rcpp

- C++ is an iteration (literally) on the C language
- Known for being extremely fast, C and C++ are typically used as benchmarks for many other languages
- Not as forgiving or as transparent as other languages. . .

Basics about C++

- Statically Typed Language
- Compiled Language
- Rcpp makes it easier for R and R Studio Users

Rcpp Workflow

- Replace only the necessary functions using C++ through Rcpp
- Export those compiled C++ functions to your R environment
- Use them just like you would an R function

Short Tutorial: Rcpp

Let's show it in action.

C/C++ for Python Users

- `Cython` is a Python package which allows C/C++ usage
- Same goal as Rcpp
- More details found at `https://cython.org/`

Another Language: Julia

- Julia is a relatively new language
- Focused on scientific applications, it focuses on being faster than C in some cases
- Far more forgiving and intuitive syntax

Using Julia in R

- The R Ecosystem does not have well vetted packages for using Julia yet...
- We will focus on using one of the best packages currently available: `JuliaCall`
- While the typical file ending for Julia files is `.jl`, using a `.julia` file ending in R Studio will enable syntax highlighting

Julia in R Workflow

- The workflow is similar to Rcpp, but slightly more manual
- Entire Julia scripts can be called from R, but it is recommend to source functions from Julia into R

Short Tutorial: JuliaCall

Let's try it out!

Julia for Python Users

- No immediately accessible libraries for cross-langauge usage
- `IJulia` package in the Julia ecosystem allows for Julia to be used in a Jupyter notebook
- More information can be found here: `https://github.com/JuliaLang/IJulia.jl`

Concluding Thoughts

- Combining Rcpp and parallelization is certainly an option!
- Combining Julia and parallelization… may be an option.
- Depending on if your task can be easily parallelized or not determines which technique used to speed up execution time

Exercises

Visit