Optimizing your code

Multiprocessing tools

Well, you are now working on a server, congrats!! so What's the big deal?! Well you have a lot of cores waiting to help you to conduct you analysis. Awesome, well wait... are not scripting languages going through my codes sequentially by default?! Don't tell me I need to rewrite all my scripts!!!!

No, you don't have to. There are low hanging fruits available to you to take advantages of multiple cores in your processing. Any for loop is an opportunity to distribute you work accross multiple cores. In R, any use of the apply family function might be an opportunity of leveraging multiprocessing. How? well as for most of the tasks there are packages and modules available to help you to do so.

R

mcapply

foreach

The foreach package provides a new looping construct for executing R code repeatedly. With moderate modification to you code, you can transform a serial execution of a for loop into a parallel execution. Here is a simple example:

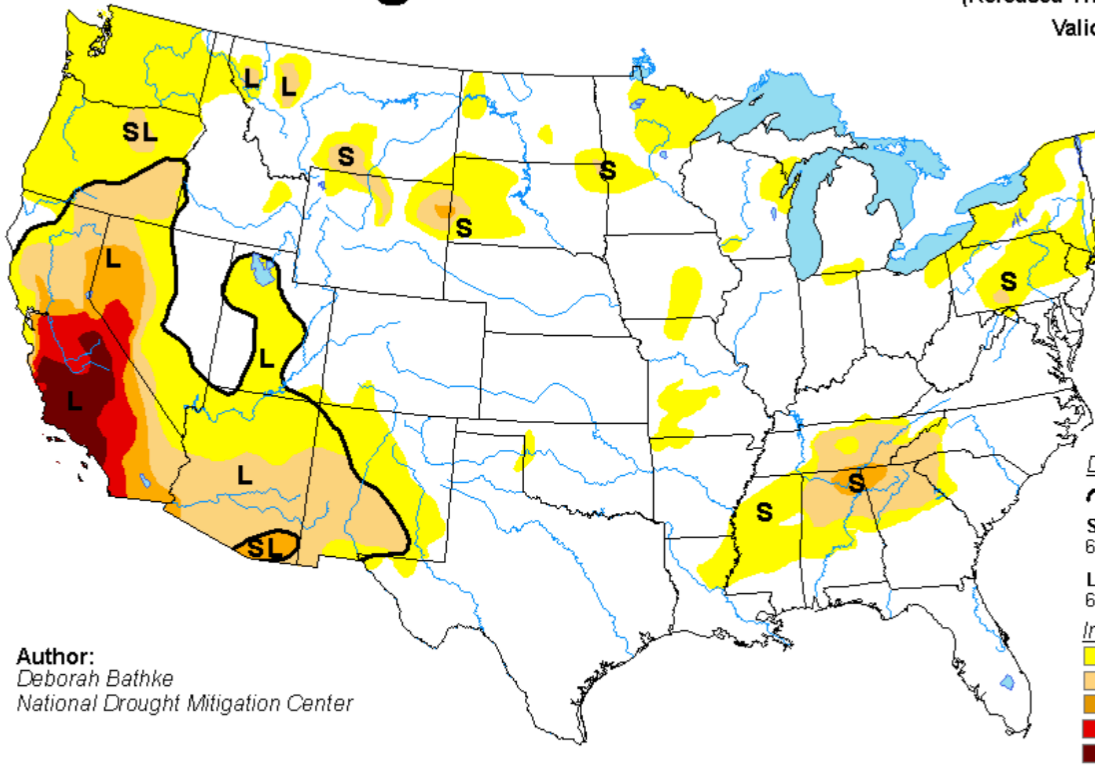
We are going to download the shapefiles produced by the United States Drought Monitor for the year 2016. There is one shapefile produced per week capturing the drought conditions over the contiguous US using a qualitative scale.

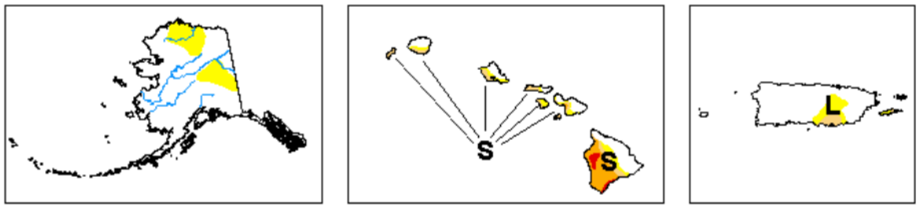
U.S. Drought Monitor

June 7, 2016

(Released Thursday, Jun. 9, 2016)

Valid 8 a.m. EDT





Drought Impact Types:

〜 Delineates dominant impacts

S = Short-Term, typically less than 6 months (e.g. agriculture, grasslands)

L = Long-Term, typically greater than 6 months (e.g. hydrology, ecology)

Intensity:

🟨 D0 Abnormally Dry
🟧 D1 Moderate Drought
🟧 D2 Severe Drought
🟥 D3 Extreme Drought
🟫 D4 Exceptional Drought

The Drought Monitor focuses on broad-scale conditions. Local conditions may vary. See accompanying text summary for forecast statements.

Author:
Deborah Bathke
National Drought Mitigation Center

http://droughtmonitor.unl.edu/

```
library(downloader)
full_url <- "http://droughtmonitor.unl.edu/data/shapefiles_m/2016_USDM_M.zip"
download(full_url,dest = zipfile_name, mode = "wb")

year_path <-
zip_files <- list.files(path=year_path, pattern = "*.zip")


# unzip several shapefiles

# Standard for loop
for (zip_file in zip_files) {
   unzip(file.path(year_path, zip_file, exdir = file.path(year_path,"SHP"), overwrite
 = True)
}
```

Python

General Notes:

- This implementation is only possible when the iterations in the for loop are independant from each others, meaning there is dependacny of the foloopwing iteration to the result of the preivous one.
- The RAM of the server is shared accross the different processes running, it can be a limitation to the number of cores you can use if you are working with large data (on one machine).
- If you have nested for loops, you generally try to parallelize the outter one

References

R

- R view: https://cran.r-project.org/web/views/HighPerformanceComputing.html
- More applied introduction to multiprocessing: http://www.glennklockwood.com/data-intensive/r/parallel-options.html
- foreach and doParallel:
    - Getting started with foreach and doParallel: https://cran.r-

    project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf
    - More info about foreach: https://cran.r-project.org/web/packages/foreach/vignettes/foreach.pdf
    - Nested for loop: https://cran.r-project.org/web/packages/foreach/vignettes/nested.pdf

- mclapply: https://stat.ethz.ch/R-manual/R-devel/library/parallel/html/mclapply.html