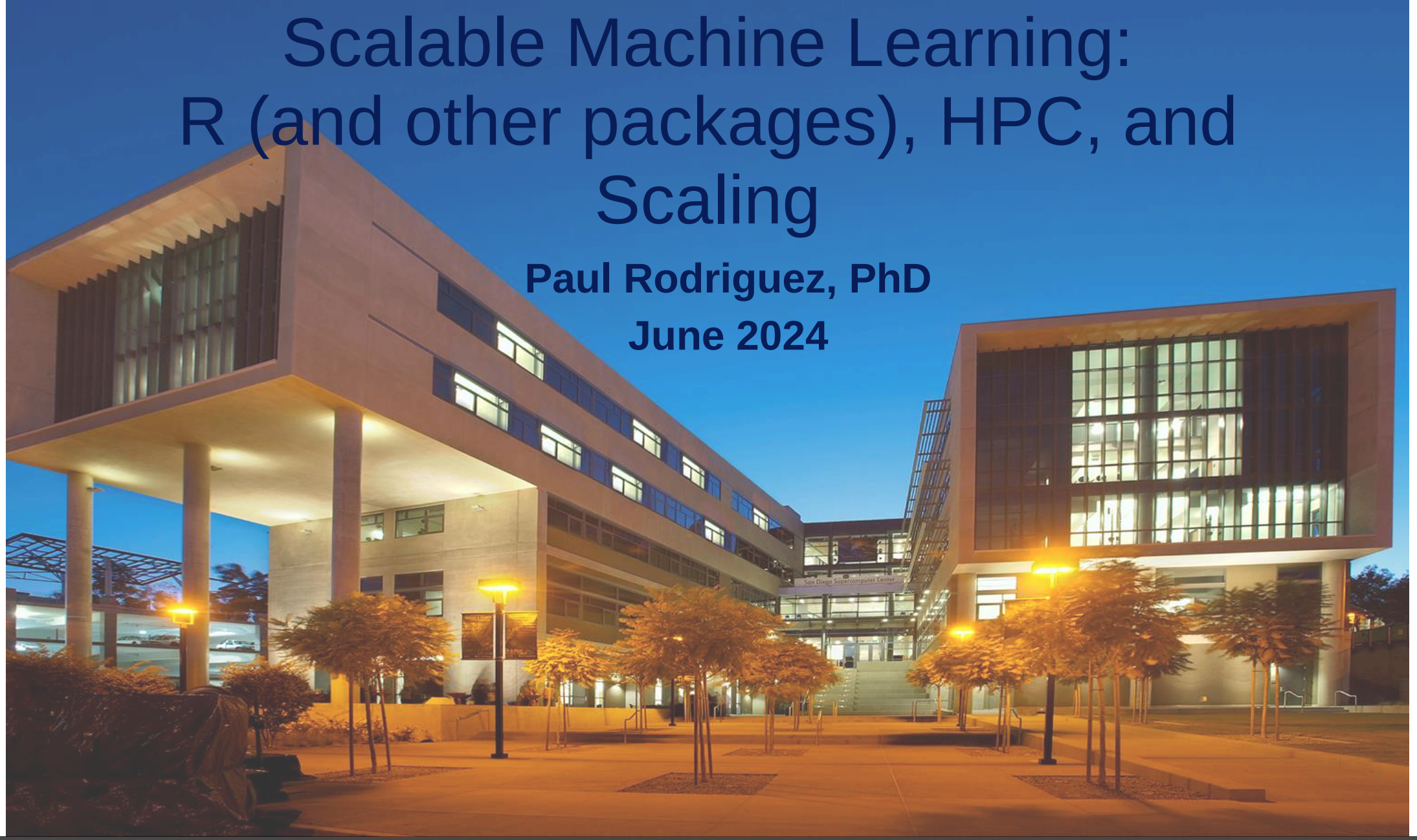
Scalable Machine Learning: R (and other packages), HPC, and Scaling

Paul Rodriguez, PhD

June 2024



Outline

- R and Scaling
- Parallel R
- Embarrassingly Parallel R
- A big data exploration of R

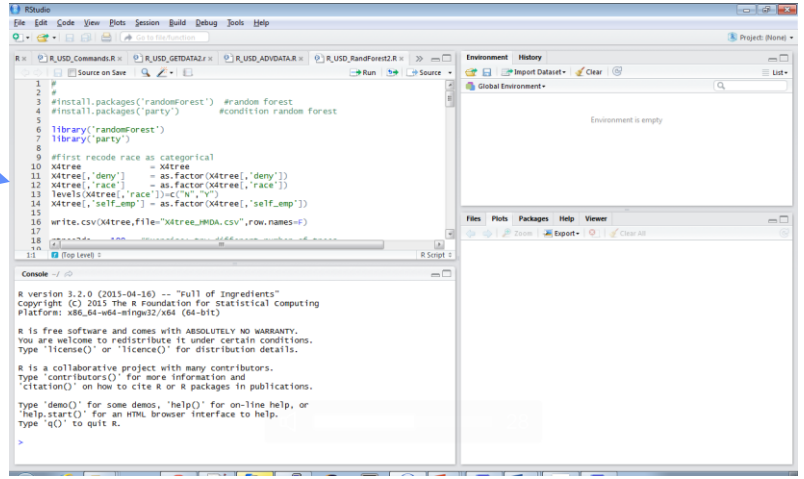
A typical R development workflow

- **R studio: An Integrated development environment for R**

Menu tab

Edit window to Build scripts

R console



Environment Information on variables and command history

Plots, help docs, package lists

Typical R code workflow

```
#READ DATA
X                   =read.csv('hmda_aer.csv',header=T,stringsAsFactors=T)

#SUBSET DATA
indices_2keep =which(X[,'s13'] %in% c(3,4,5)))
X                   =X[unique(indices_2keep),]

#CREATE/TRANSFORM VARIABLES
pi_rat              = as.numeric(X[,'s46']/100)          #debt2income ratio


#RUN MODEL and SHOW RESULTS
lm_result           =lm(deny~pi_rat)            #lm is ‘linearmodel’
summary(lm_result)
```

R strengths for HPC (IMHO)

- **Data Wrangling –**

- **Particular statistical procedure implementations -**
  Imputation methods (for missing data)
  Sampling methods
  Instrument Variable (2 stage) Regression
  Matching subjects for pairwise analysis
  MCMC routines (but Stan is likely better package)
  Generalized Linear Model
  Some ML model (e.g. randomForest)

R Scaling In a nutshell

- R uses BLAS/LAPACK math libraries for operations on vectors

  [Same for Matlab and Python]

- R packages provide multicore, out-of-core, multinode, or distributed data (SparkR) options

  [Same for Matlab and Python]

- Some ML model implementations may be built to use parallel backends (review the available options)

R Scaling In a nutshell

- **R sessionInfo() command shows math libraries**

```
> sessionInfo()
R version 4.0.2 (2020-06-22)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: CentOS Linux 8 (Core)

Matrix products: default
BLAS/LAPACK: /cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen2/gcc-9.2.0/openblas
grrcfayp3br6kmcnelbgrepqmadwv43e/lib/libopenblas_zenp-r0.3.10.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
```

Consider Regression Computations

- **Linear Model:** $Y = X * B$

  where Y=outcomes , X=data matrix

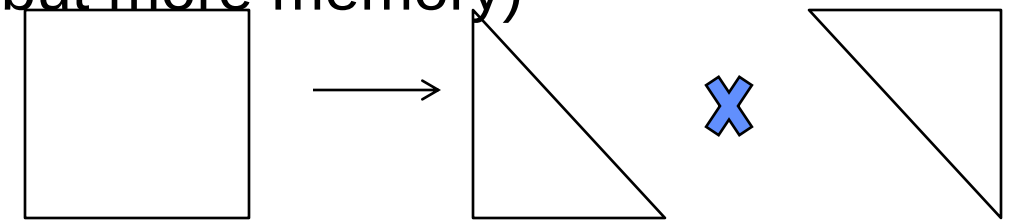
- **Algebraically, we could**:

  take "inverse" $of\ X * Y = B$ (time consuming)

  use derivatives to search for solutions (very general)

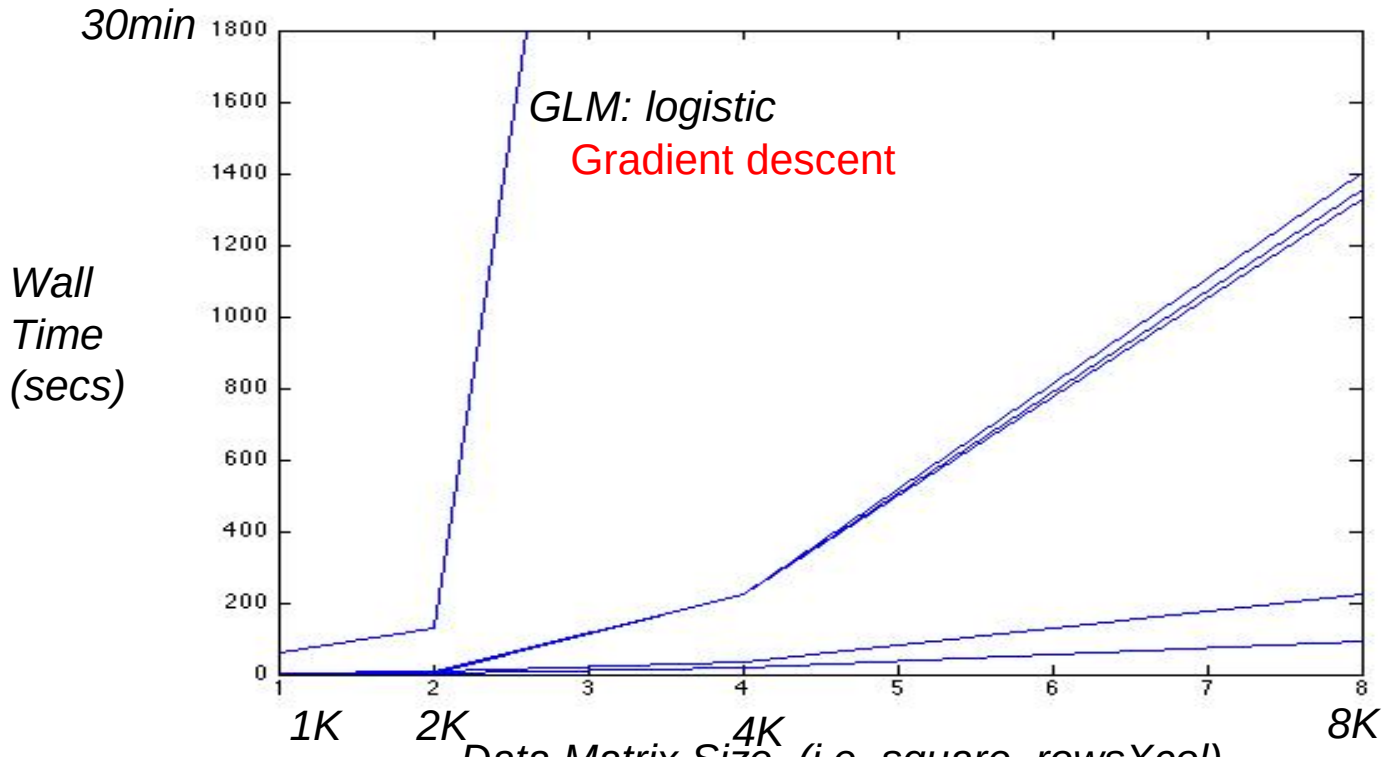
- **Or, better:**

  QR decomposition of *X* into triangular matrices (easier to solve but more memory)



Solving Linear Systems
Performance with R, 1 compute node

R:
glm(Y~X,family=gaussian) #gaussn regrssn (like lm)
glm(Y~X,family=binomial) # logistic regrssn (Y=0 or 1)



Matrix inverse solutions

glm: Gaussian
lm()
Inverse()

QR solutions

Solve(a,b)
QR

R multicore processing

- ‘doParallel’ package – provides the back end to the ‘for each’ parallel processing command
- uses threads across cpu cores to pass data & commands
- It also works for multinode (runs on top of RMPI)

See https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)
```

1. allocate workers

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(
```

1. allocate workers

2. Make ‘foreach’ loop

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(i=1:24,.combine=rbind)
```

1. allocate workers

2. Make ‘foreach’ loop

3. specify how to combine results

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(i=1:24,.combine=rbind) %dopar%
      {   …
```

1. allocate workers

2. Make ‘foreach’ loop

3. specify how to combine results

4. %dopar% runs it across cores, (%do% runs it serially)

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(i=1:24,.combine=rbind) %dopar%
      {   …
            your code here

          return(  a variable or object )
    })
```

1. allocate workers

2. Make ‘foreach’ loop

3. specify how to combine results

4. %dopar% runs it across cores, (%do% runs it serially)

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(i=1:24,.combine=rbind) %dopar%
      {   …
            your code here

          return(  a variable or object )
    })
```

1. allocate workers

2. Make ‘foreach’ loop

3. specify how to combine results

4. %dopar% runs it across cores, (%do% runs it serially)

BEWARE: foreach will copy data to every core if its seems necessary

R multinode: parallel backend

```
library(doParallel)

cl <- makeCluster(48)
registerDoParallel(cl)
```

1. allocate cluster as parallel backend

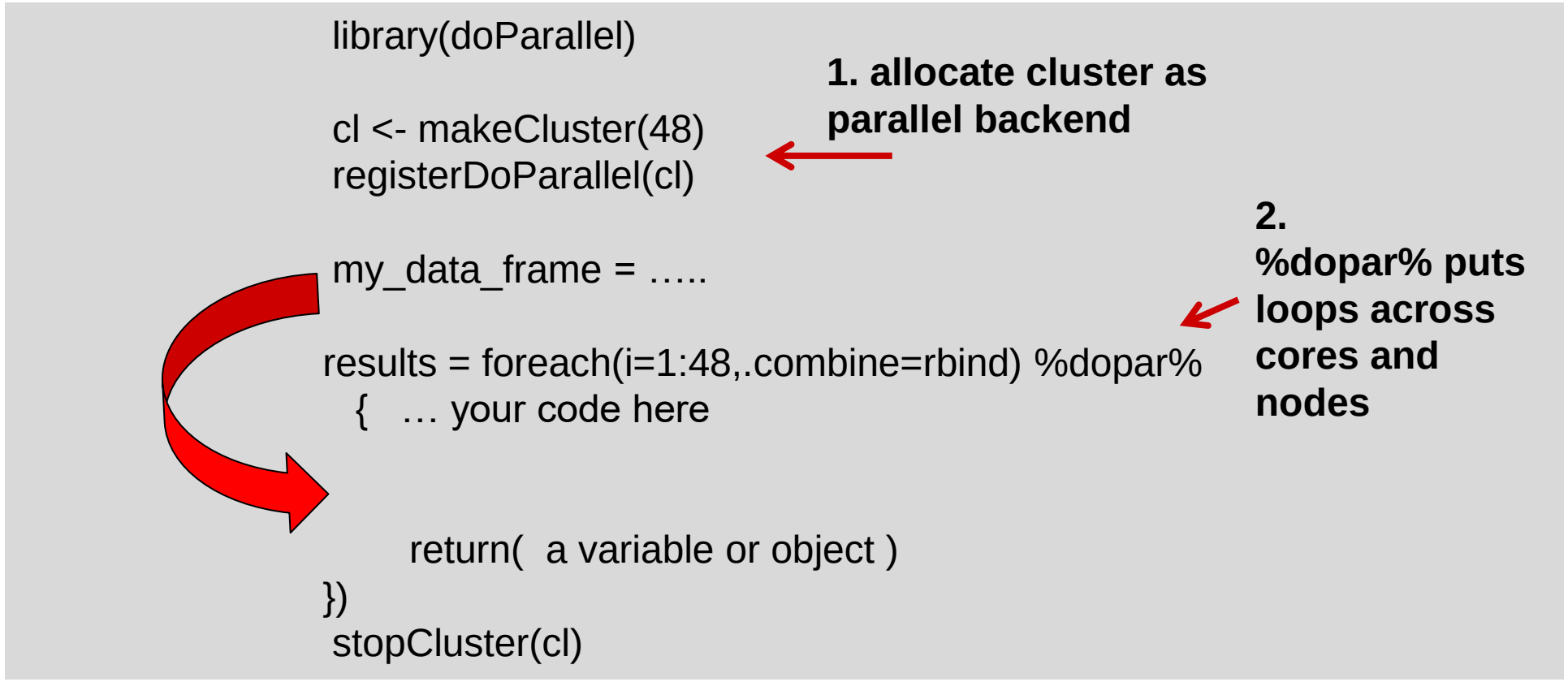
R multinode: parallel backend

```
library(doParallel)

cl <- makeCluster(48)
registerDoParallel(cl)

my_data_frame = …..

results = foreach(i=1:48,.combine=rbind) %dopar%
  {   … your code here


       return(  a variable or object )
})
 stopCluster(cl)
```

1. allocate cluster as parallel backend

2. %dopar% puts loops across cores and nodes

R multinode: parallel backend



BEWARE: foreach will copy data to every core in every node if its seems necessary

Speaking of R and memory…

Recall the other day Marty noticed, someone was running R (using $top)

Several R processing going, and only 16Gb free memory left!

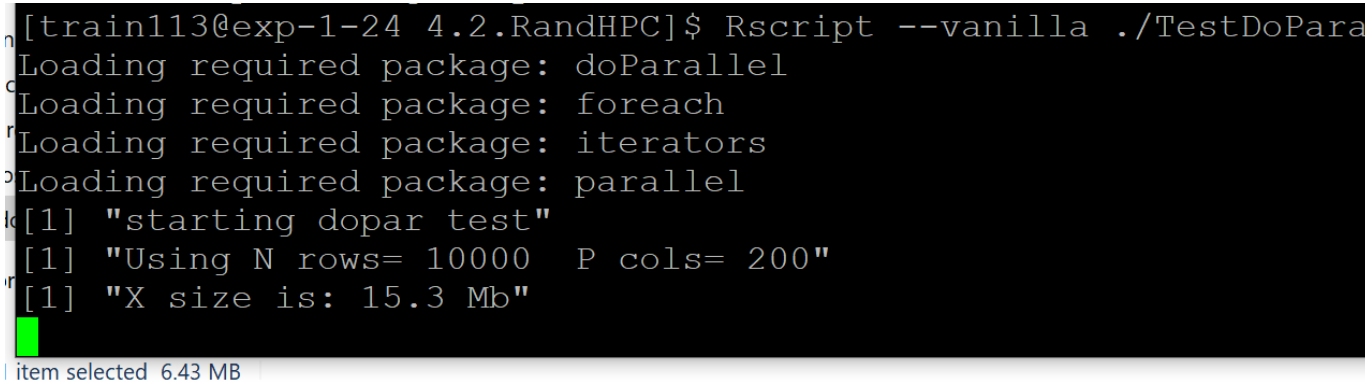
```
top - 16:27:58 up 35 days, 22:29, 234 users,  load average: 211.08, 147.10, 87.3
Threads: 7388 total,  25 running, 7118 sleeping, 242 stopped,   3 zombie
%Cpu(s):  5.8 us,  7.5 sy,  0.0 ni, 67.4 id, 19.0 wa,  0.1 hi,  0.1 si,  0.0 st
MiB Mem : 127842.9 total,  15550.3 free,  61423.8 used,  50868.8 buff/cache
MiB Swap:  12288.0 total,    244.0 free,  12044.0 used.  64421.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
3170165 mrincon   20   0    7016   3448   1540 R  99.9   0.0  14:48.71 gzip
3430379 keyajos+  20   0  529984  26628  13900 R  99.7   0.0  21:05.47 cpptraj
3503080 keyajos+  20   0  528032  26388  14052 R  99.7   0.0   8:13.02 cpptraj
   3563 root      20   0 9881.4m 138724 135408 S  43.8   0.1 17638:46 in:imjo+
2979421 jis038    20   0 7968716  31444  11536 R  41.0   0.0  34:23.16 R
2979433 jis038    20   0 7968716  31444  11536 R  41.0   0.0  34:21.34 R
2979432 jis038    20   0 7968716  31444  11536 R  40.3   0.0  34:21.18 R
2979435 jis038    20   0 7968716  31444  11536 R  40.3   0.0  34:25.71 R
2979431 jis038    20   0 7968716  31444  11536 R  40.0   0.0  34:22.04 R
3531483 sgolzari  20   0 2437332   1.6g  17492 R  12.7   1.2   0:37.56 conda
1965101 sgolzari  20   0 7928092  27712   5752 R  11.7   0.0  31:53.05 R
 397946 sgolzari  20   0 7909624  26136   4788 R  11.1   0.0  60:16.22 R
1965093 sgolzari  20   0 7928092  27712   5752 R  11.1   0.0  31:49.91 R
 397945 sgolzari  20   0 7909624  26136   4788 R  10.8   0.0  60:17.28 R
 397947 sgolzari  20   0 7909624  26136   4788 R  10.8   0.0  60:18.83 R
 827069 sgolzari  20   0 7909628  26468   5360 R  10.8   0.0  51:36.85 R
```

Testing/Evaluating R parallel command line

- **Exercise: 'TestdoParallel' R script**

1. Log into expanse portal
2. $ srun-compute
3. $ cd ciml..institute../4.2_r_on_HPC folder
4. $ Rscript –vanilla TestDoParallel_v1.R

(this might first install DoParallel R package and ask for confirmation and repo site – choose a US site)



Also start a 2nd terminal and ssh into that compute node and run $ top –u $USER - how's memory usage?

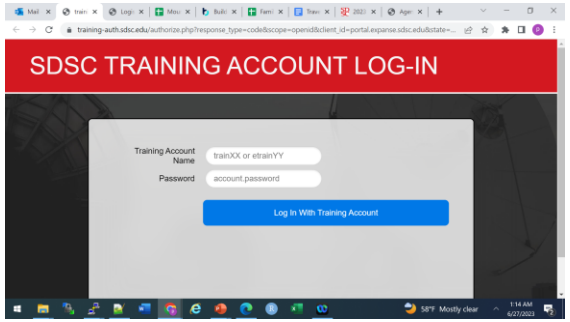
In top, enter H to see threads, enter f -> down arrow -> space -> esc to toggle cupid

Testing/Evaluating R parallel in portal

- **Exercise: 'TestdoParallel' R script**

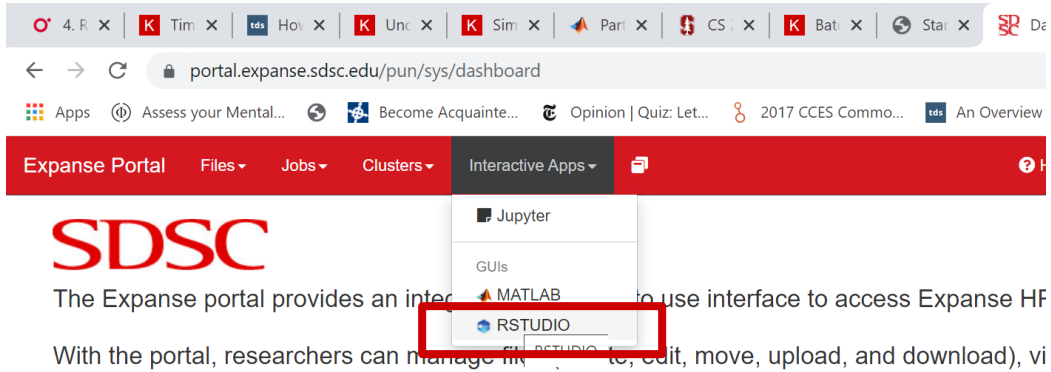
1. **Log into expanse portal and start R studio**

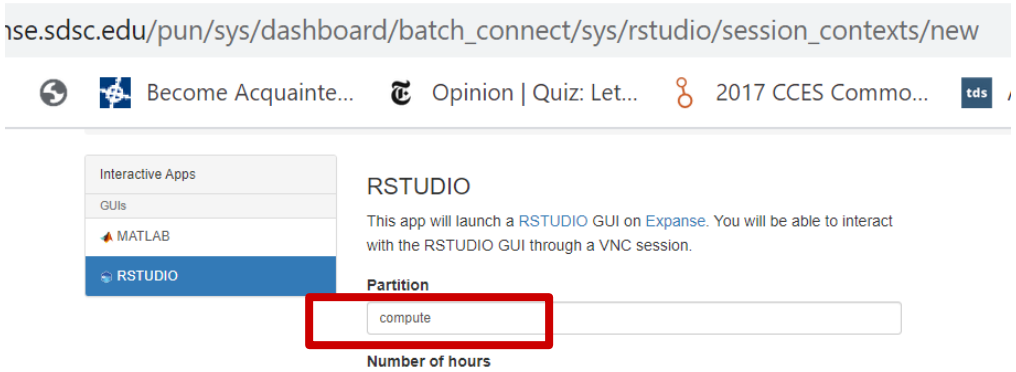
   goto URL: *https://portal.expanse.sdsc.edu/training*

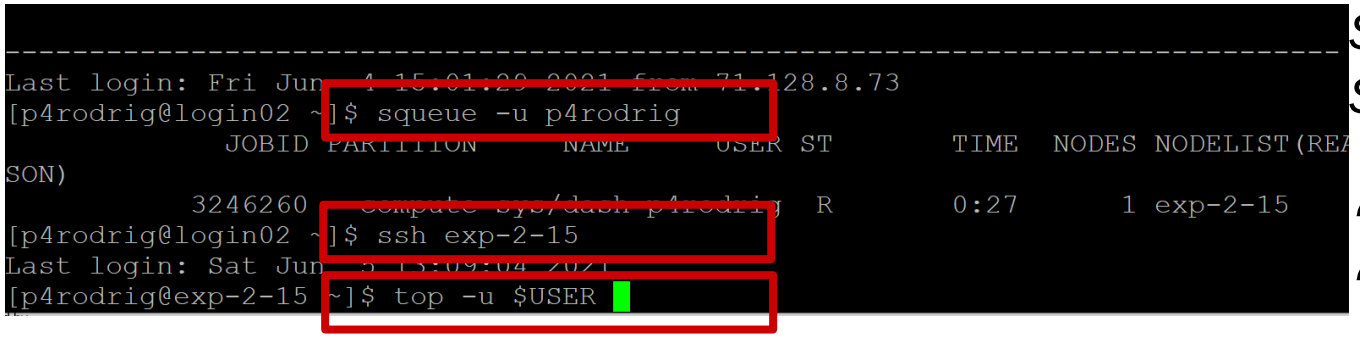


2. **Also log into expanse command line and ssh to compute node**

3. **run 'top –u username' to see performance**
   - look for tradeoffs in memory vs execution as matrix size varies (see next slides)







1 Open portal ->
Interactive Apps ->
Rstudio

Enter
Node: "compute"
Cores: "64"
Memory: 124 Gb
(other fields defaults ok)

2 Also login to Expanse terminal window

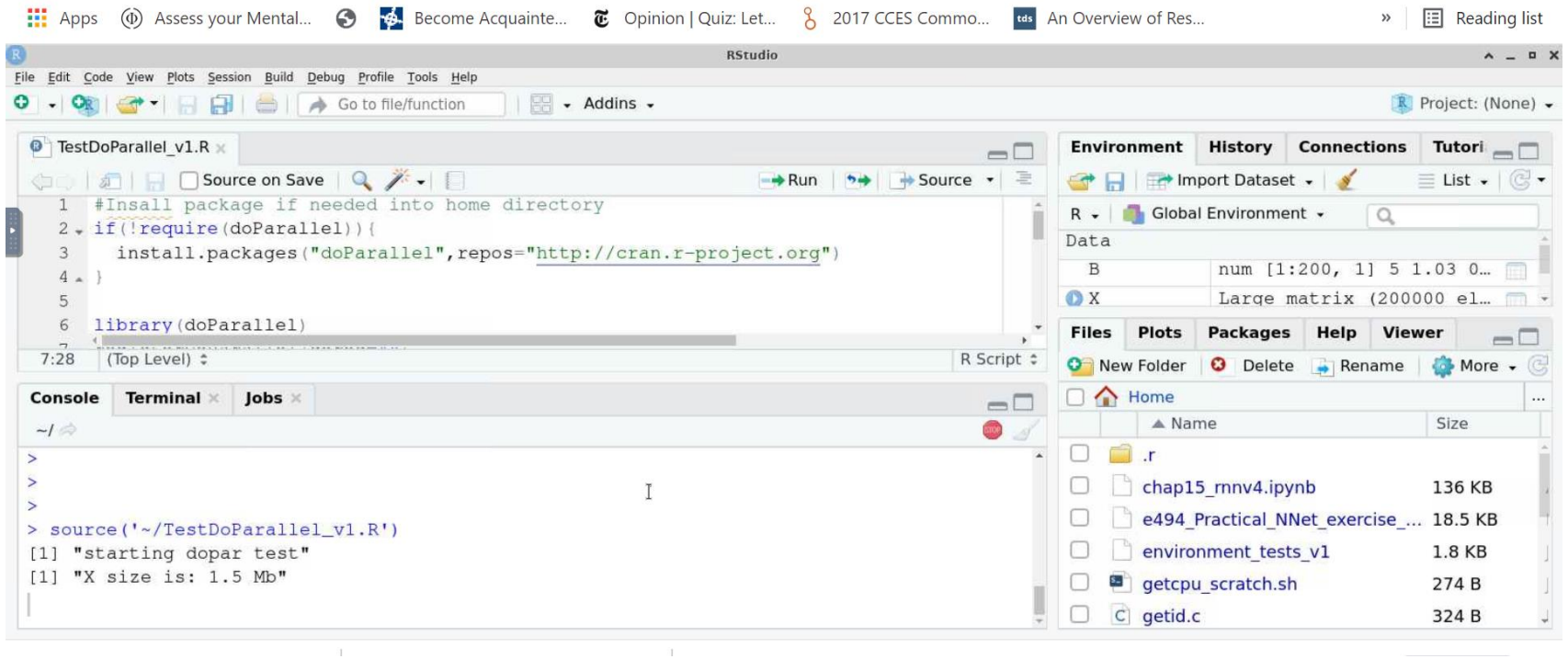
$ squeue –u $USER
$ ssh exp-##-##
$ top –u $USER

'H' will toggle threads
'f' , downarrow to P, space, esc.

3 Open the 'Test_doParallel ' Rscript

Select 'source' to run the whole script, it will install 'doParallel' package (if the R installation doesn't have it already)

look for # <<< ----- comments to change data parameters

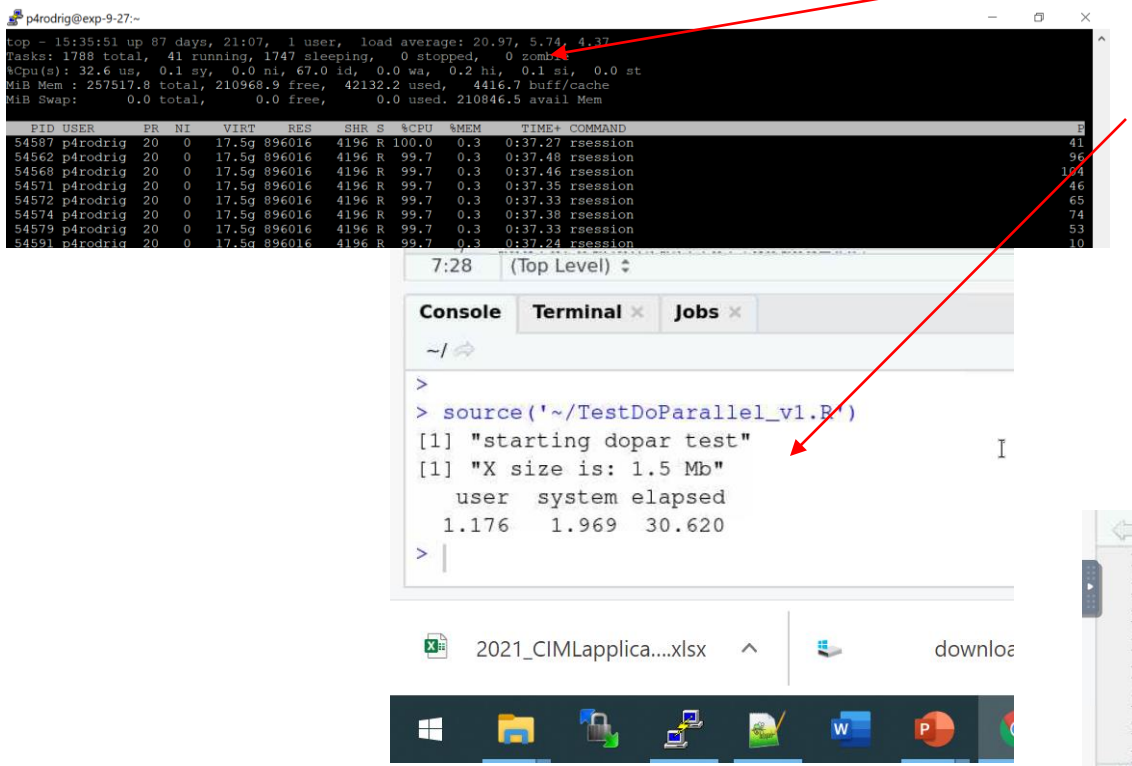


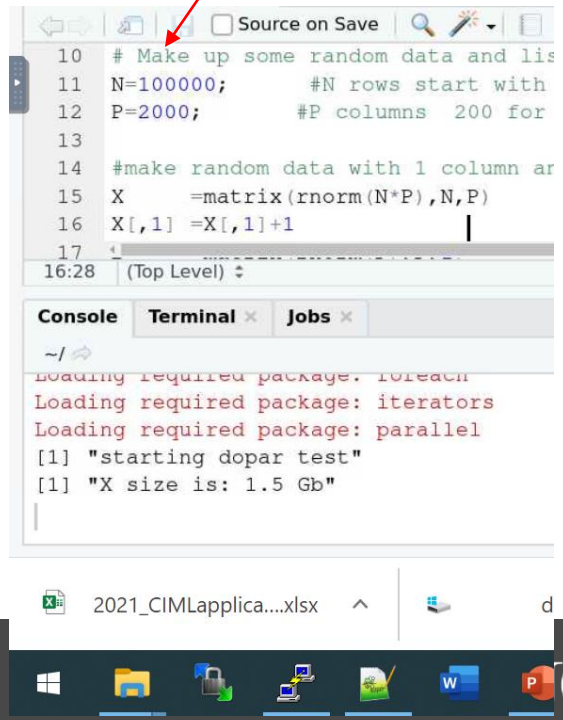
Review the top output

Notice the elapsed time and memory size

Change the NxP matrix size and rerun

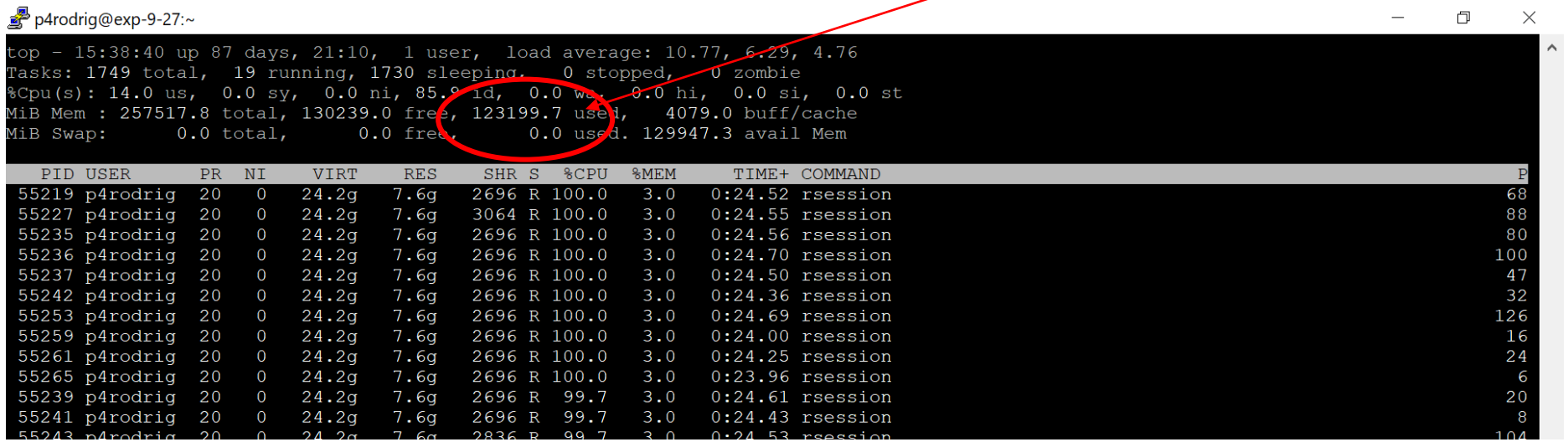
(start with N=10K, P=2K)





Try this at home:

Let N=100K , P=2000
Notice the memory used is close to 124Gb we asked for



If you ask for 248Gb will it run?
What if you use only 24 cores?

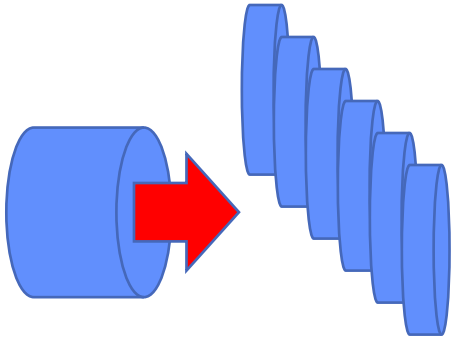
Parallezing for loops

(pseudo code)

R with doParallel	Matlab with parallel toolbox	Python with dask.distributed
makecluster *registercluster* *foreach with dopar,* *combine results*	*parcluster('local')* *parpool()* *parfor* *or* *'spmd' with distributed arrays* gather array	Import delayed, Client Client(numwkrs) for i in range(numwkrs): A=delayed(my_func)(i) Acombine.append(A) Acombined.compute()

An option for (embarrassingly) Parallel R

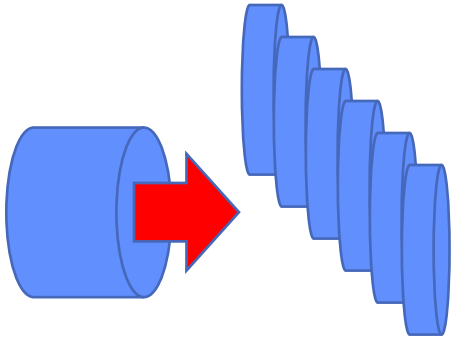
1. Split up data into N parts

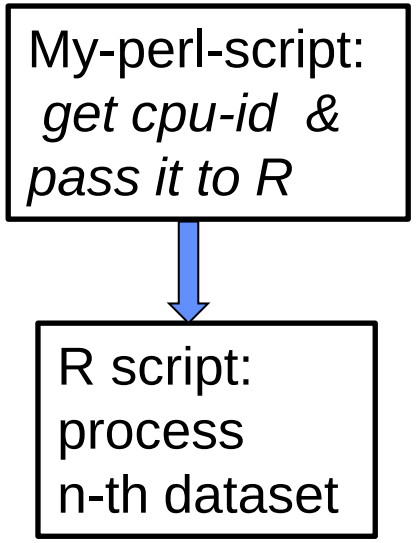


An option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:
   mpirun … my-perl-script

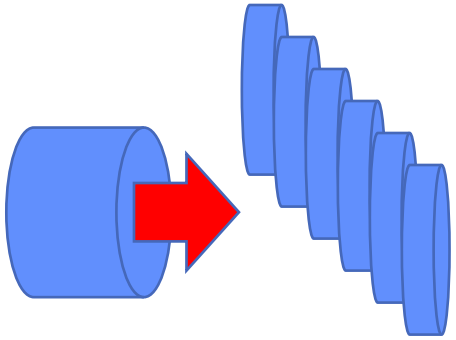


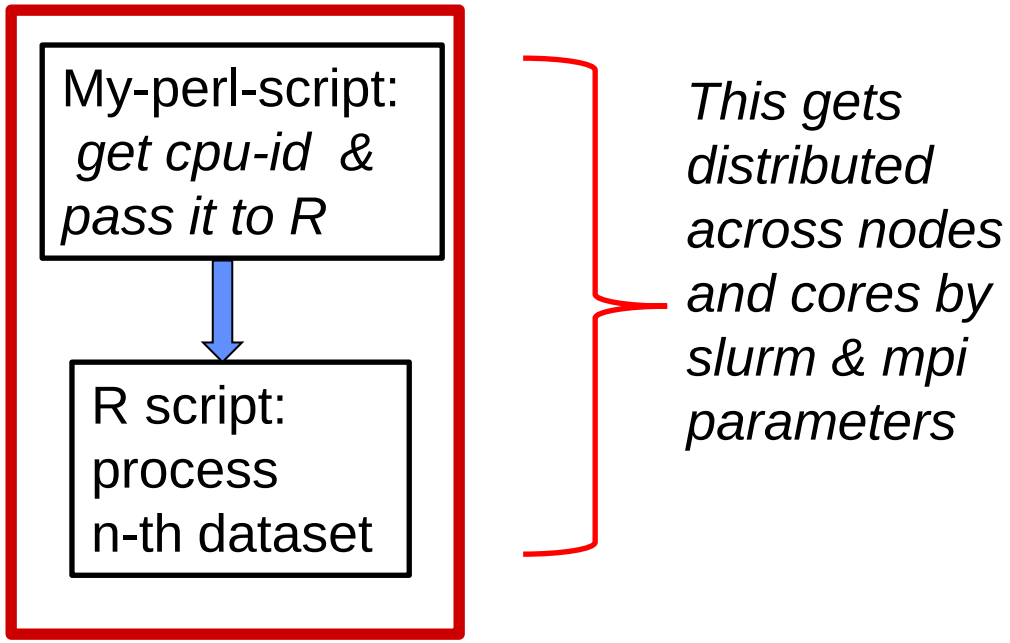


An option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:
   mpirun … my-perl-script





Slurm parameters: one R instance per core across all nodes

Normal batch job info

```
…
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=128
#SBATCH --cpus-per-task=1

module load slurm
module load cpu
module load gcc
module load intel-mpi

mpirun -genv I_MPI_PIN_DOMAIN=omp:compact ./get_mpirank_runcmd.pl
```

2 x128 = 256 mpi ranks

256 perl script/R instances
1 core each

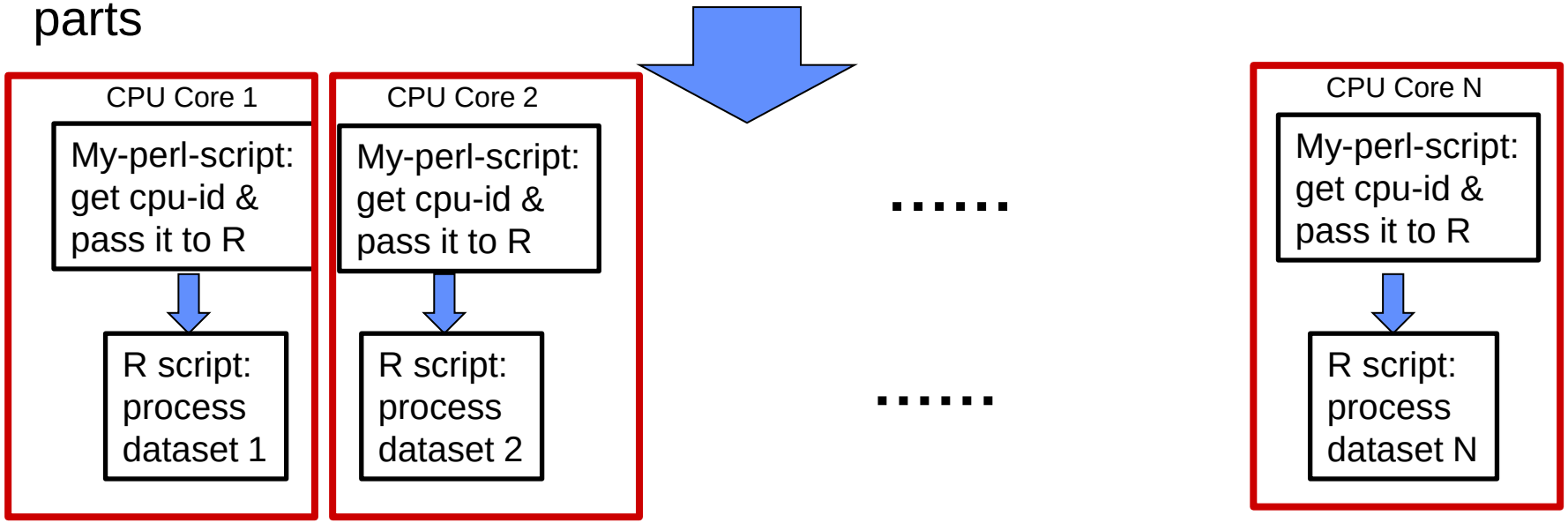
(based on /cm/shared/examples/sdsc/mpi-openmp-hybrid/hybrid-slurm.sb)

one R instance per core across all nodes

1. Split up data into N parts

2. In slurm batch script:

mpirun … my-perl-script

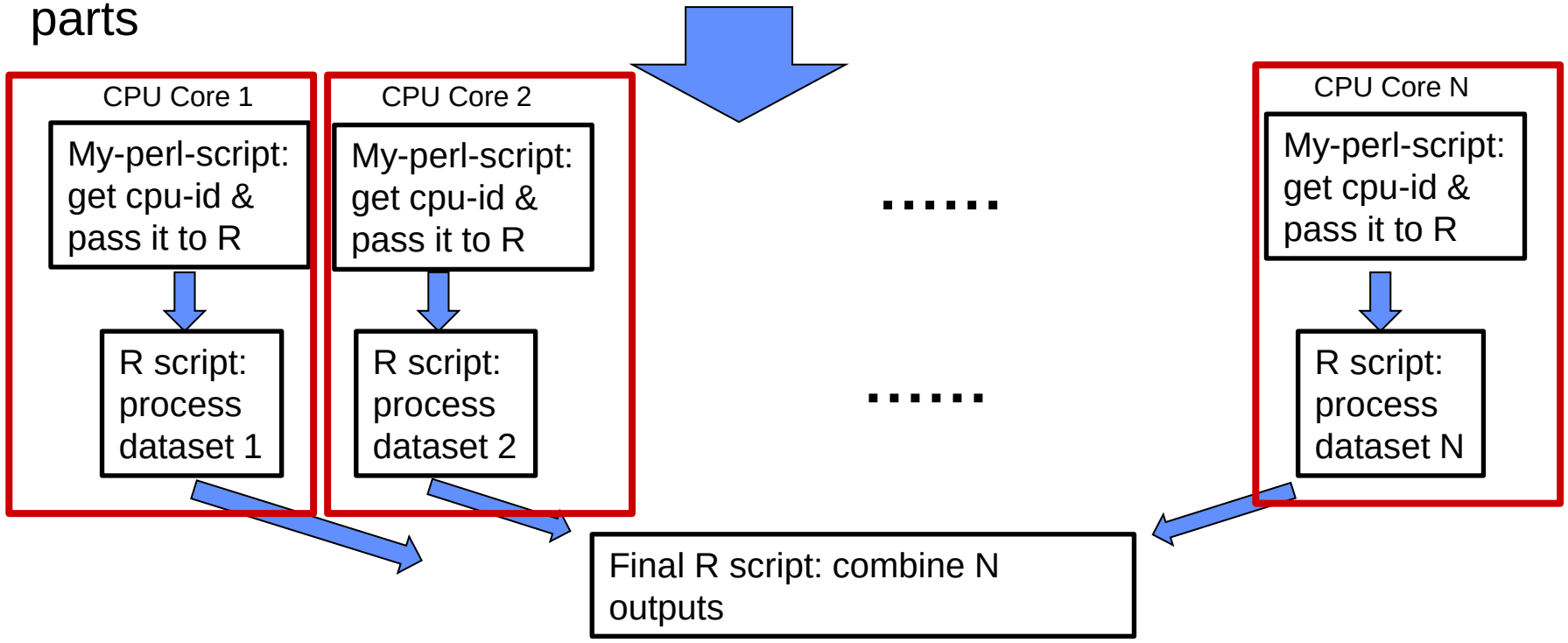


one R instance per core across all nodes

1. Split up data into N parts

2. In slurm batch script:

mpirun … my-perl-script



More programming but perhaps more useful

Slurm parameters: one R instance per node with 128 cores per R instance

Normal batch job info

```
…
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=128

module load slurm
module load cpu
module load gcc
module load intel-mpi

module load r

mpirun -genv I_MPI_PIN_DOMAIN=omp:compact ./get_mpirank_runcmd.pl
```

2 x1 = 2 mpi ranks

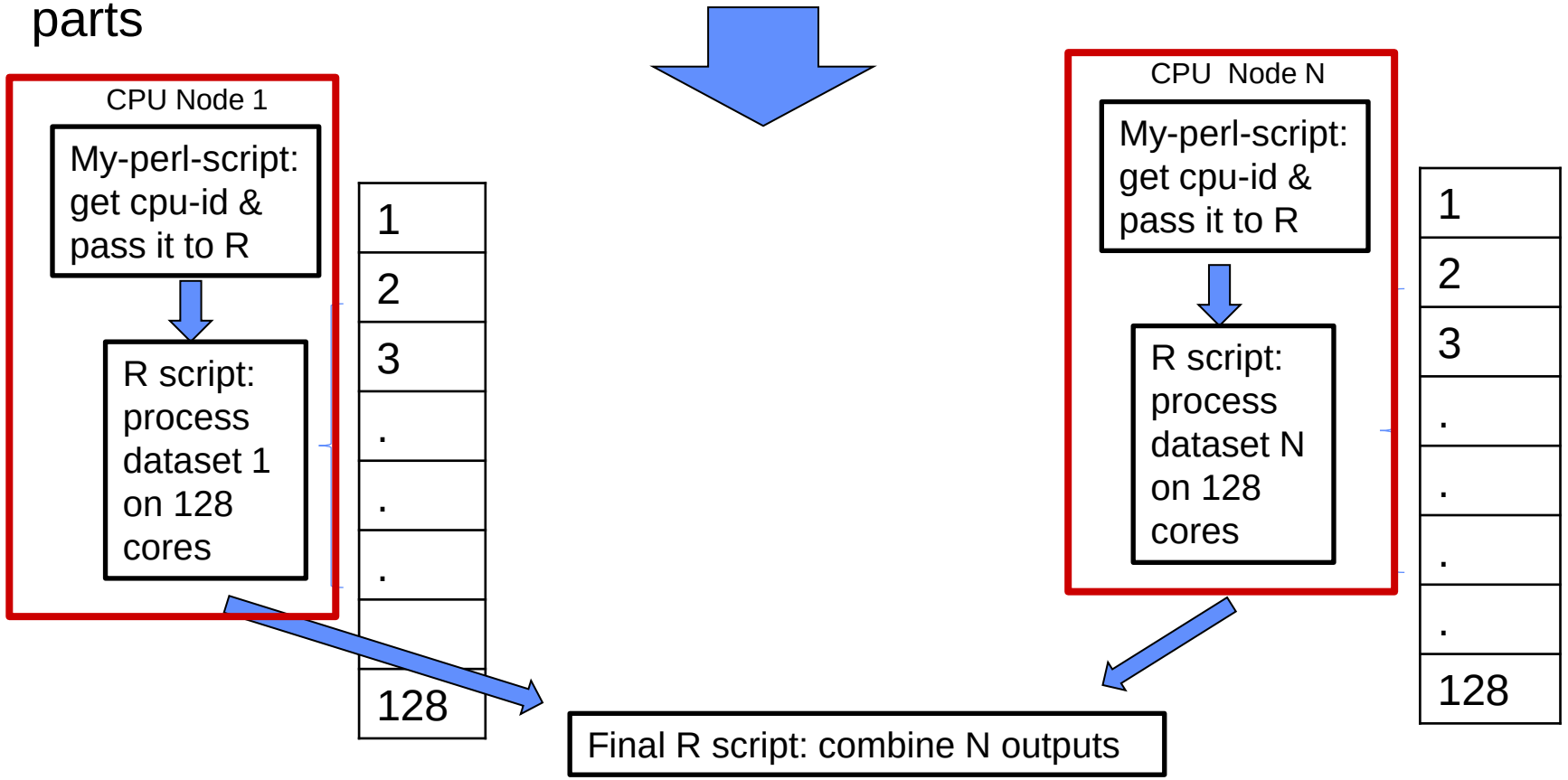
2 perl script/R instances
128 cores each
(doParallel can use them)

Example: One R instance per node, doParallel across all cores in each node

1. Split up data into N parts

2. In slurm batch script:

mpirun … my-perl-script



Big Data exploration

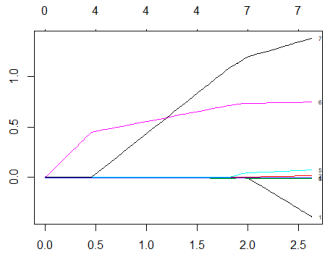
- Run R ‘biglasso’ with a dataset too big for RAM
- Create large CSV file (117Gb) of X data matrix and Y outcomes:
  Y=X*B + noise (where X is 100K x50K)
- Explore other packages with out-of-core dataset functions:
  Matlab, Dask-ML, Spark, Keras
- Beware: lasso implementations can differ;
  also, Y,X,B should be ‘nice’

Lasso Regression

- **Penalized Loss function** $L = \boldsymbol{MSE} + \boldsymbol{\lambda} \sum |b_i|$

- **Recall: using penalty term is the same as using a constraint (constrained optimization)**

$$\textbf{find min } \boldsymbol{MSE} \; such\; that \sum |bi| < S$$



- **Different implementations may:**
  - use different fit methods (ie forward stepwise, coordinate descent, gradient descent,…)
  - might parallelize by splitting up data, computations, or vectorizing
  - read/load data more or less efficiently

Considerations

- All packages have special functions to handle out of core dataset
- Sometimes better to have Y,X together; sometimes two files are better

- All were run as Expanse batch jobs, not through notebook/portal

- All were set up run on 1 compute node (248Gb RAM) and use scratch (ssd) space. (Large memory nodes are available on Expanse, but not tested)

- Mostly default parameters used; Little optimization performed; Not a benchmark study!

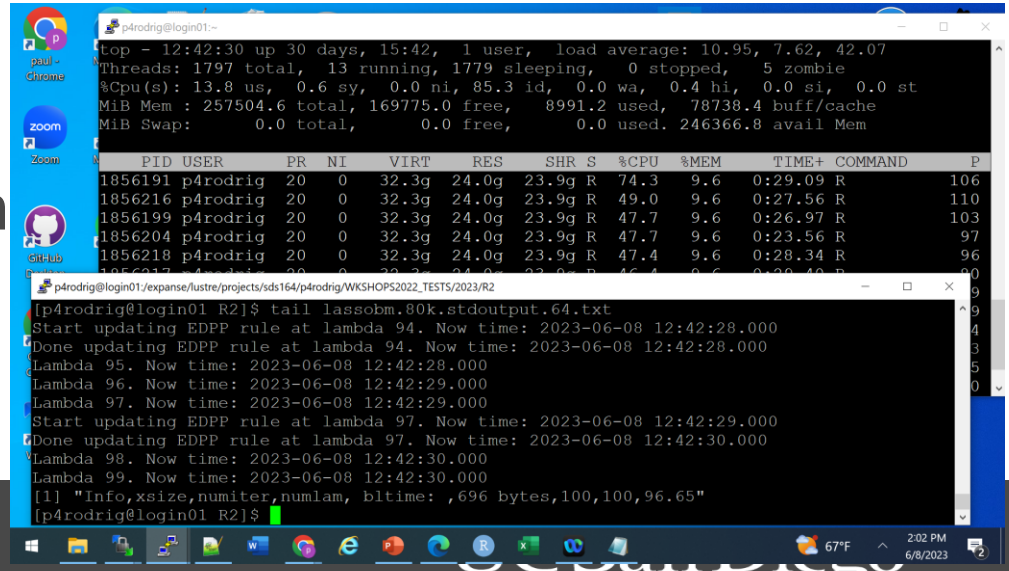
R

- R – biglasso (bigmatrix) package to set up file backed dataframe

https://cran.rstudio.com/web/packages/biglasso/

Issue: the file backend for the big data object path option was hard to get right – ended up just running out of scratch SSD as working directory;

Outcome: R copies everything into binary file an descriptor file and got results in about 2hours



R code highlights

Use biglasso package

Y data fits in memory so just read it in

X data will be setup with file-backed memory

biglasso() arguments look like the glmnet implementation

```
library(biglasso)
….
Y.bm=read.big.matrix(inputYfile,sep = ",")
X.bm=setupX(inputXfile,sep=",",
                        type="double",
                        backingfile     = "x.bin",
                        descriptorfile = "x.desc")
….

bl_results=biglasso( X.bm, Y.bm,
                              row.idx = 1:nrow(X.bm),
                              penalty = c("lasso"),
                              family = c("gaussian"),
                              ncores = numcores,   …
```

Batch script for R biglasso, highlights

…..

#SBATCH --partition=compute
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=64
#SBATCH –A account
… etc

module load slurm
module load r
…etc

#copy data to scratch
cp Xmatrix_100000_50000.csv /scratch/$USER/job_$SLURM_JOB_ID/Xinput.csv
cp Ymatrix_100000_1.csv /scratch/$USER/job_$SLURM_JOB_ID/Yinput.csv

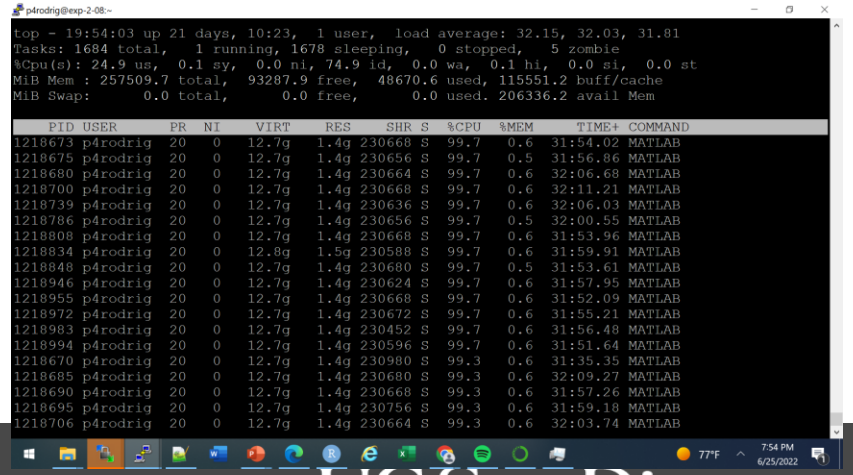
Rscript --vanilla ***path-2-your-script/rLassov2.R***
/scratch/$USER/job_$SLURM_JOB_ID/Xinput.csv
/scratch/$USER/job_$SLURM_JOB_ID/Yinput.csv 64 TRUE > save_std_output.txt

Matlab

- Matlab – has tall table functions to read in large CSV file; but lasso needs tall array

I found a ‘table2array’ function that helped set up the data

But matlab tall arrays are not intended to run with too much memory, or too many columns, b/c lasso uses correlation matrix



Dask-ml

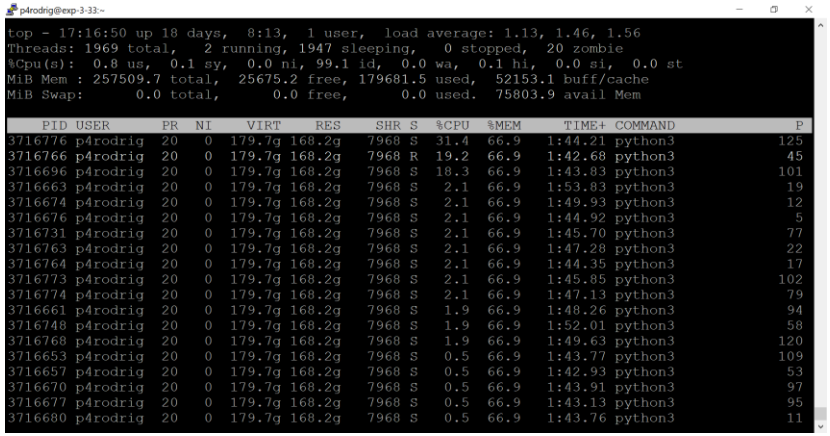
- Dask-ML – worked easily with distributed dataframe, built on top of sklearn,

read_csv uses a ‘blocksize’ parameter

repeatedly calls sklearn ‘partial-fit’ functions that iterate 1 time for ‘batch’ processing

worked better with Y&X in one dataframe

I stopped it after 4hrs, but it was running



Spark

- **Spark – uses distributed dataframe with pyspark.ml.regression package**

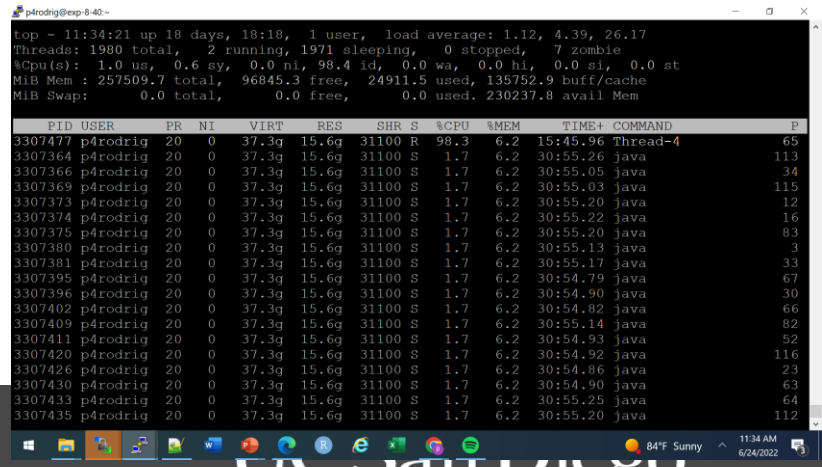
Use the vector assembler options to get Y and ‘features’ all in one dataframe

Got some help on spark session options:

device & executor memory, eg: local[64], 8Gb, 2Gb

I stopped it after 4 hrs; it was running

Local[64], 50 min in, 1 iteration



Keras

- **Keras - set up a tensorflow dataset with data generator (to reading batches at a time from csv file); a little tricky to get shapes right;**

I set up a 1 layer linear neural network with a L1 regularizer on weights

runs, 1000 epochs in 1 hour+ (on cpu)

I did not set up convergence options like other Lasso implementations

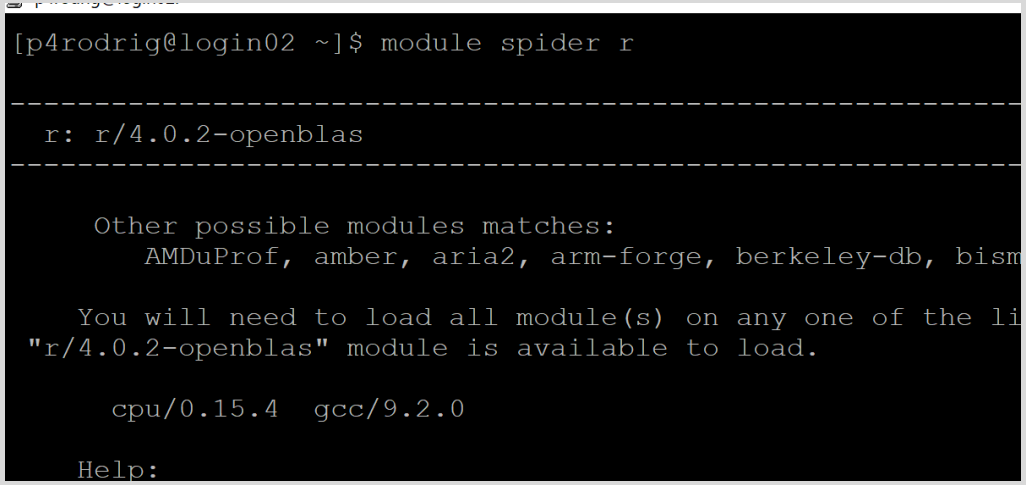
Some notes

- Start with small data with interactive session or notebook – maybe even just use a smaller sample?
- All packages generally work as documented, but
- All packages require working through some implementation issues or environment options for the session/job/execution

How to use R directly on Expanse

1. Get an interactive compute node:

2. Try
$ module spider r *(this tells you what modules you need)*



3. Enter
$ module load cpu/0.15.4
$ module load gcc/9.2.0
$ module load r/4.0.2-openblas

$ R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
…..
Type 'q()' to quit R.

>

A note on installing R Packages (into your own directories)

- **In R (might help to be on interactive node):**

  install.packages('package-name')

  (see https://cran.r-project.org/ for package lists and reviews)

- **Sometimes you have to be explicit:**

  install.packages('ggmap',
  repos='http://cran.us.r-project.org',dependencies=TRUE)

  If compiling is required and you get an error, call support

  Packages are put into your /home/user/R directory

Other R package possibilities:

- **Rspark - R interface to Spark**
- **R Keras – R interface to Keras**
- **pdbR - distributed matrix support (better for dense matrices vs Spark)**
- **Rgputools – GPU support**

THE END