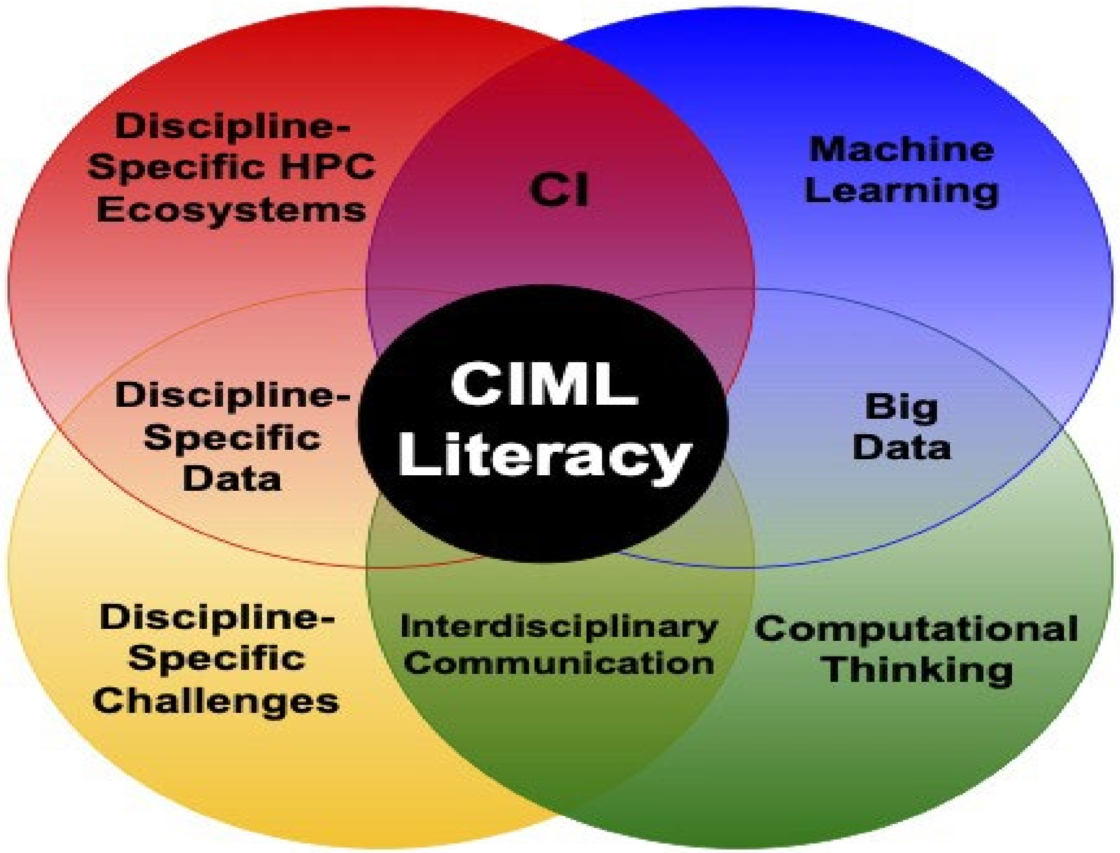
Scalable Machine Learning: R (and other packages), HPC, and Scaling

July 2022

Paul Rodriguez, PhD
(SDSC)


Discipline-Specific HPC Ecosystems
CI
Machine Learning
Discipline-Specific Data
CIML Literacy
Big Data
Discipline-Specific Challenges
Interdisciplinary Communication
Computational Thinking

Outline

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

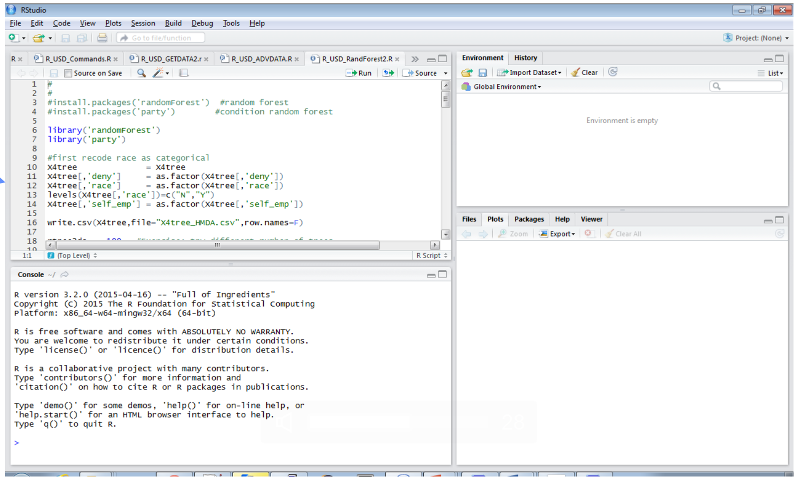
A typical R development workflow

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

Menu tab

Edit window to Build scripts

R console



```
#
#
#install.packages('randomForest')  #random forest
#install.packages('party')         #condition random forest

library('randomForest')
library('party')

#first recode race as categorical
X4tree                = X4tree
X4tree[,'deny']       = as.factor(X4tree[,'deny'])
X4tree[,'race']       = as.factor(X4tree[,'race'])
levels(X4tree[,'race'])=c("N","Y")
X4tree[,'self_emp'] = as.factor(X4tree[,'self_emp'])

write.csv(X4tree,file="X4tree_HMDA.csv",row.names=F)
```

```
R version 3.2.0 (2015-04-16) -- "Full of Ingredients"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

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]

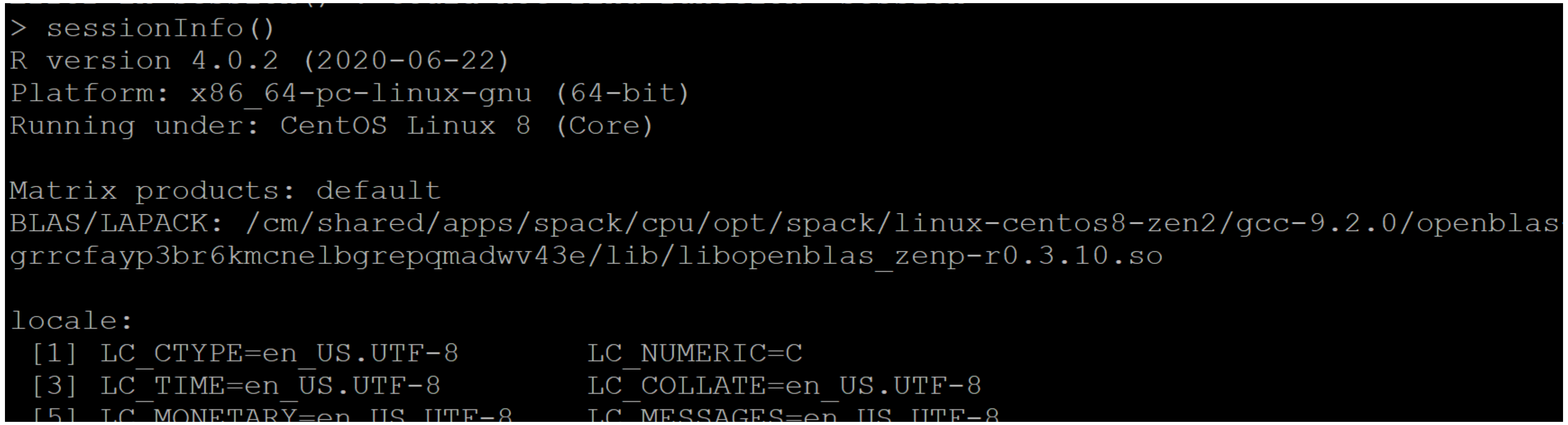
- R packages provide multicore, out-of-core, multinode, or distributed data (SparkR) options

  [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

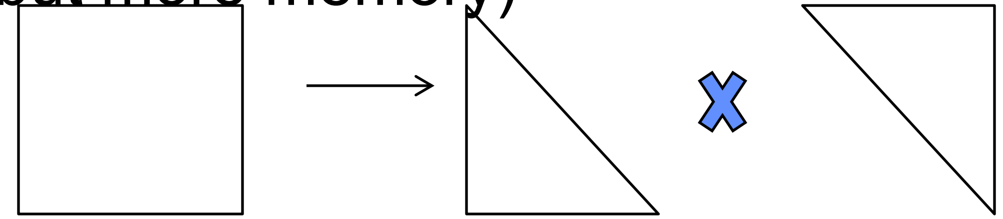
- **Algebraically, we could**:

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

  use derivatives to search for solutions (very general)

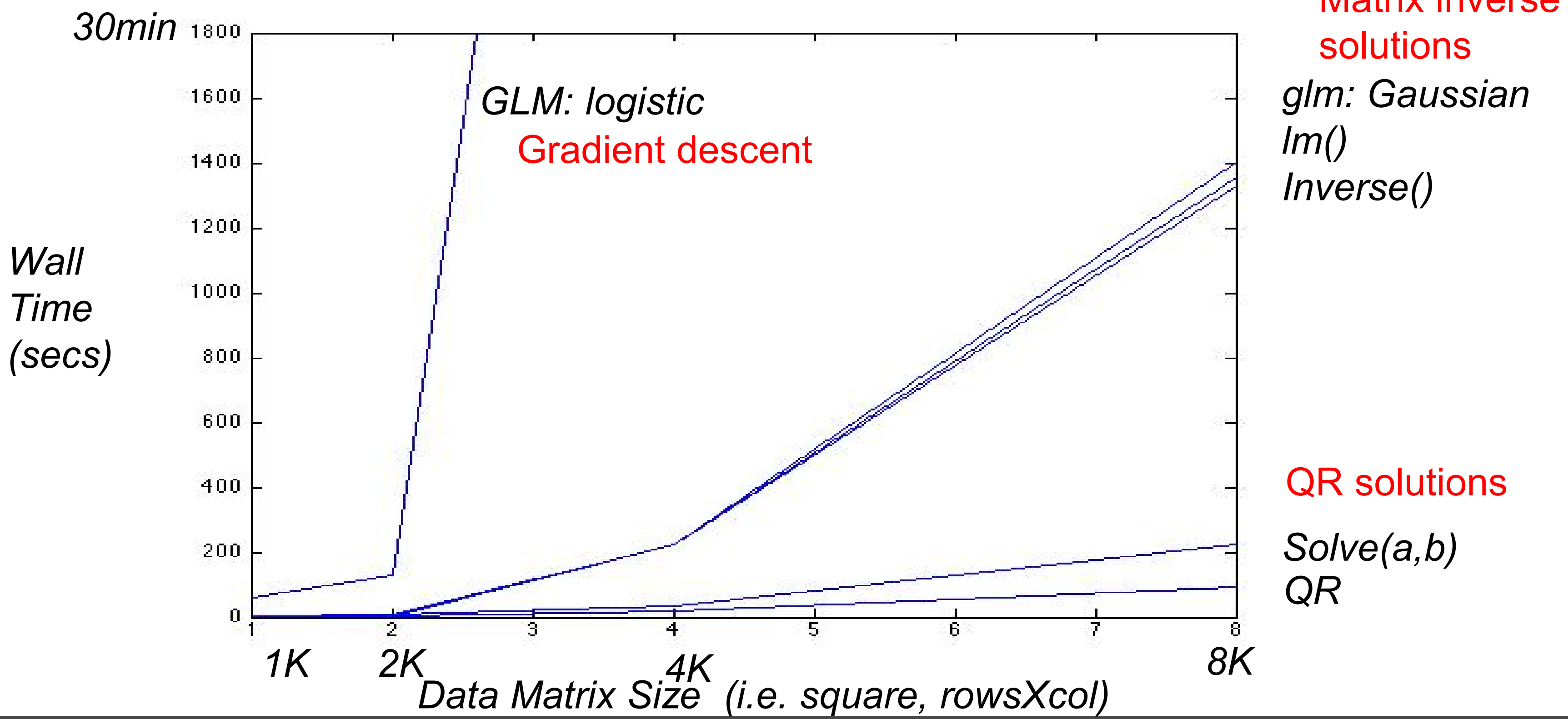
- **Or, better:**

  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)



Outline

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

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)
```

$\leftarrow$ **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

```
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

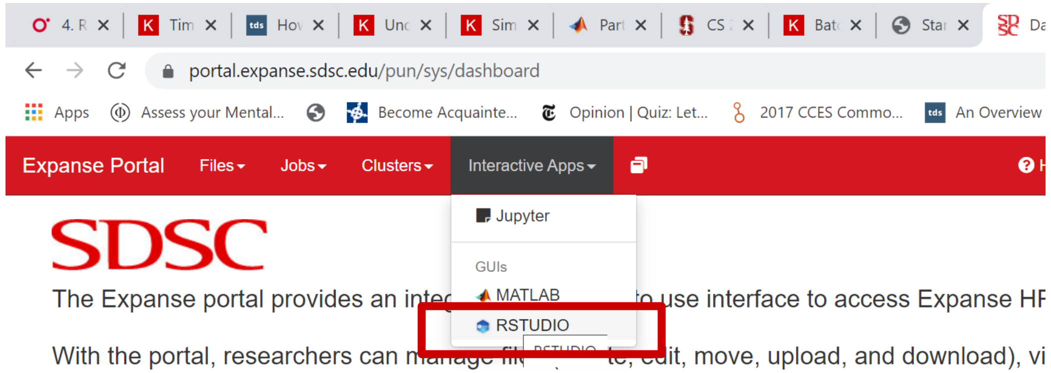
2. %dopar% puts loops across cores and nodes

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

Testing/Evaluating R parallel

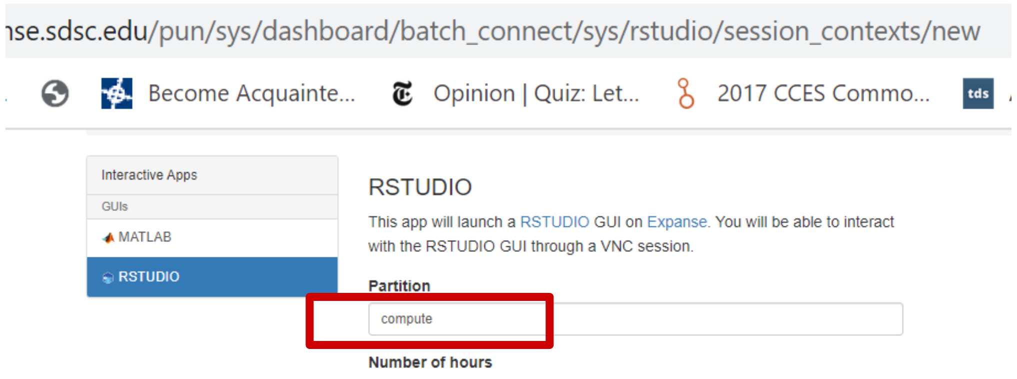
- **Exercise: ‘TestdoParallel’ R script**

1. **Log into expanse portal and start R studio**
2. **Also log into expanse command line and ssh to compute node**
3. **run ‘top –u username’ to see performance**
   - Enter ‘H’ to see threads; Enter f, toggle ‘P’ to see cpu ids
   - look for tradeoffs in memory vs execution as matrix size varies (see next slides)



The Expanse portal provides an integ... to use interface to access Expanse HP

With the portal, researchers can manage fil... te, edit, move, upload, and download), vi



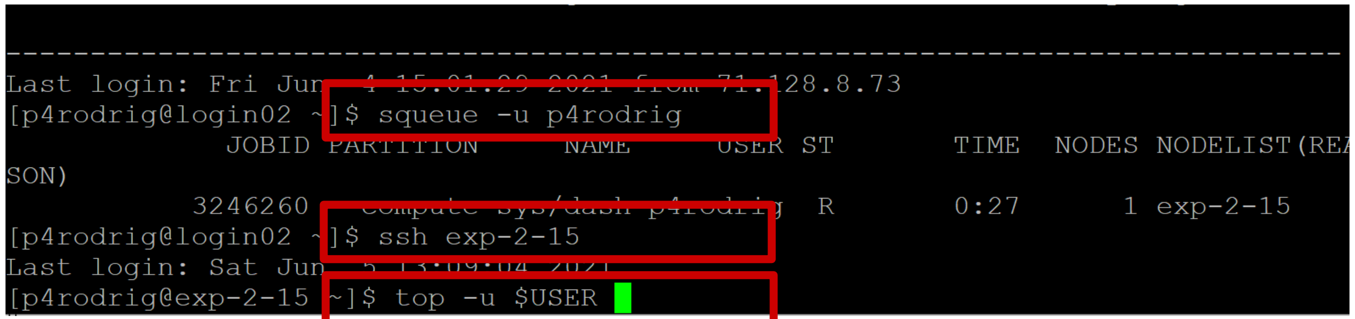
RSTUDIO

This app will launch a RSTUDIO GUI on Expanse. You will be able to interact with the RSTUDIO GUI through a VNC session.

Partition

compute

Number of hours



```
--------------------------------------------------------------------------------
Last login: Fri Jun  4 15:01:29 2021 from 71.128.8.73
[p4rodrig@login02 ~]$ squeue -u p4rodrig
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REA
SON)
           3246260   compute sys/dash p4rodrig  R       0:27      1 exp-2-15
[p4rodrig@login02 ~]$ ssh exp-2-15
Last login: Sat Jun  5 13:09:04 2021
[p4rodrig@exp-2-15 ~]$ top -u $USER
```

1

Open portal ->
Interactive Apps ->
Rstudio

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

2

Also login:
login.expanse

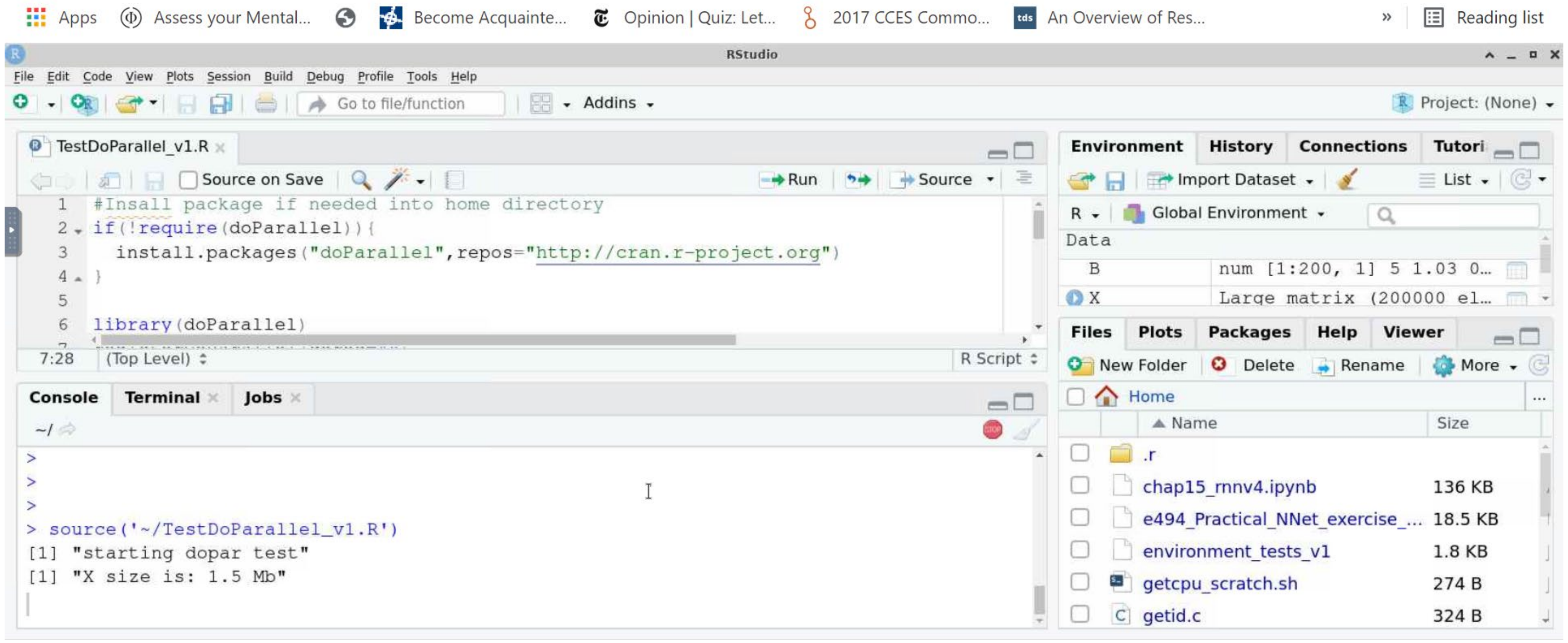
$ squeue –u $USER
$ ssh exp-##-##
$ top –u $USER

'H' will toggle threads

'f' , highlight P, space, esc will toggle last cpuid

Open the 'Test_doParallel ' Rscript

Select 'source' to run the whole script



```
#Insall package if needed into home directory
if(!require(doParallel)){
  install.packages("doParallel",repos="http://cran.r-project.org")
}

library(doParallel)
```

```
>
>
>
> source('~/TestDoParallel_v1.R')
[1] "starting dopar test"
[1] "X size is: 1.5 Mb"
```

Data

B	num [1:200, 1] 5 1.03 0...
X	Large matrix (200000 el...

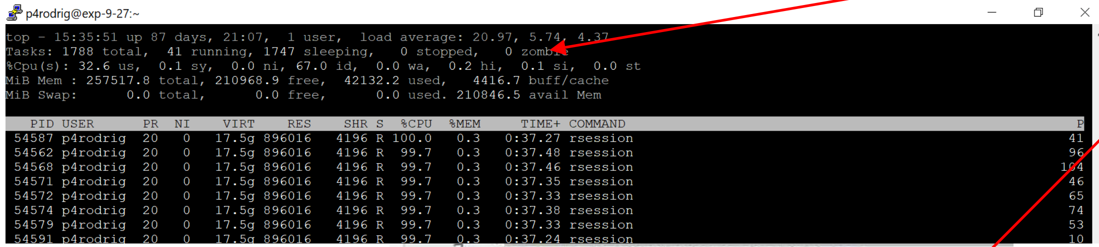
Name	Size
.r	
chap15_rnnv4.ipynb	136 KB
e494_Practical_NNet_exercise_...	18.5 KB
environment_tests_v1	1.8 KB
getcpu_scratch.sh	274 B
getid.c	324 B

Review the top output

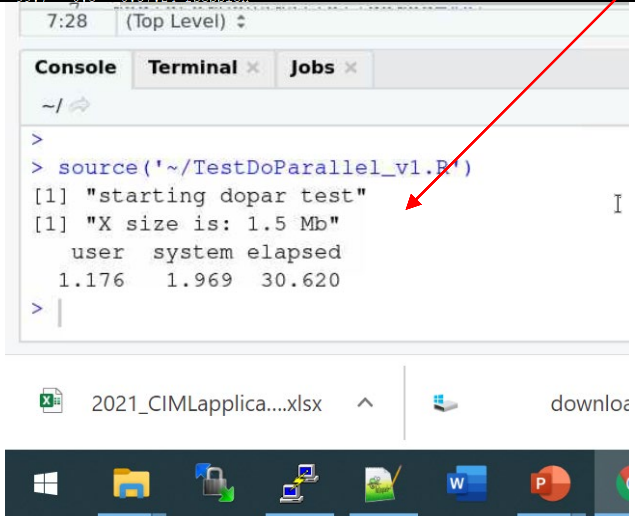
Notice the elapsed time and memory size

Change the NxP matrix size and rerun

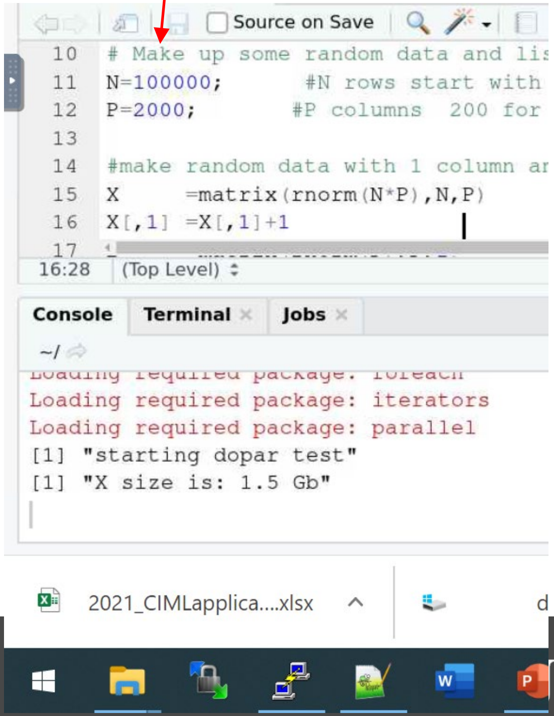
(start with N=10K, P=2K)



```
top - 15:35:51 up 87 days, 21:07,  1 user,  load average: 20.97, 5.74, 4.37
Tasks: 1788 total,  41 running, 1747 sleeping,   0 stopped,   0 zombie
%Cpu(s): 32.6 us,  0.1 sy,  0.0 ni, 67.0 id,  0.0 wa,  0.2 hi,  0.1 si,  0.0 st
MiB Mem : 257517.8 total, 210968.9 free,  42132.2 used,   4416.7 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 210846.5 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND                                         P
54587 p4rodrig  20   0   17.5g 896016   4196 R 100.0   0.3   0:37.27 rsession                                       41
54562 p4rodrig  20   0   17.5g 896016   4196 R  99.7   0.3   0:37.48 rsession                                       96
54568 p4rodrig  20   0   17.5g 896016   4196 R  99.7   0.3   0:37.46 rsession                                      104
54571 p4rodrig  20   0   17.5g 896016   4196 R  99.7   0.3   0:37.35 rsession                                       46
54572 p4rodrig  20   0   17.5g 896016   4196 R  99.7   0.3   0:37.33 rsession                                       65
54574 p4rodrig  20   0   17.5g 896016   4196 R  99.7   0.3   0:37.38 rsession                                       74
54579 p4rodrig  20   0   17.5g 896016   4196 R  99.7   0.3   0:37.33 rsession                                       53
54591 p4rodrig  20   0   17.5g 896016   4196 R  99.7   0.3   0:37.24 rsession                                       10
```



```
>
> source('~/TestDoParallel_v1.R')
[1] "starting dopar test"
[1] "X size is: 1.5 Mb"
   user  system elapsed
  1.176   1.969  30.620
>
```

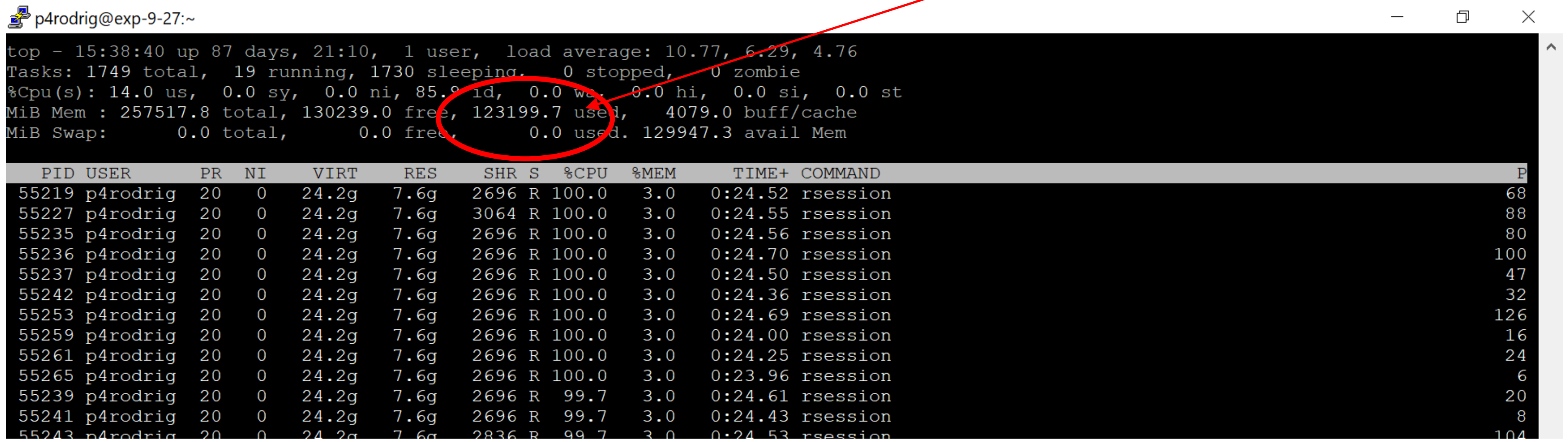


```
# Make up some random data and lis
N=100000;        #N rows start with
P=2000;          #P columns  200 for

#make random data with 1 column ar
X       =matrix(rnorm(N*P),N,P)
X[,1]  =X[,1]+1
```

```
Loading required package: foreach
Loading required package: iterators
Loading required package: parallel
[1] "starting dopar test"
[1] "X size is: 1.5 Gb"
```

Try this at home:

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



p4rodrig@exp-9-27:~

```
top - 15:38:40 up 87 days, 21:10,  1 user,  load average: 10.77, 6.29, 4.76
Tasks: 1749 total,  19 running, 1730 sleeping,   0 stopped,   0 zombie
%Cpu(s): 14.0 us,  0.0 sy,  0.0 ni, 85.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 257517.8 total, 130239.0 free, 123199.7 used,   4079.0 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 129947.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND                                  P
55219 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.52 rsession                                68
55227 p4rodrig  20   0   24.2g   7.6g   3064 R 100.0   3.0   0:24.55 rsession                                88
55235 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.56 rsession                                80
55236 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.70 rsession                               100
55237 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.50 rsession                                47
55242 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.36 rsession                                32
55253 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.69 rsession                               126
55259 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.00 rsession                                16
55261 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.25 rsession                                24
55265 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:23.96 rsession                                 6
55239 p4rodrig  20   0   24.2g   7.6g   2696 R  99.7   3.0   0:24.61 rsession                                20
55241 p4rodrig  20   0   24.2g   7.6g   2696 R  99.7   3.0   0:24.43 rsession                                 8
55243 p4rodrig  20   0   24.2g   7.6g   2836 R  99.7   3.0   0:24.53 rsession                               104
```

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

Parallezing for loops

(pseudo code)

R with doParallel

```
makecluster
registercluster

foreach with dopar,

combine results
```

Matlab with parallel toolbox

```
parcluster('local')
parpool()

parfor
or
'spmd' with
distributed arrays

gather array
```

Python with dask.distributed

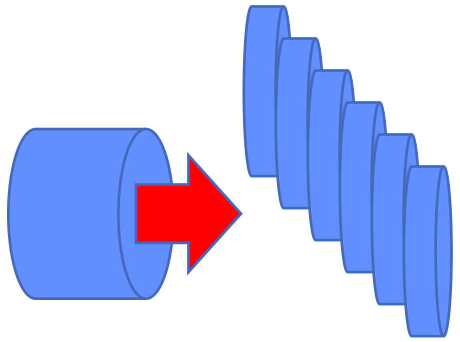
```
Import delayed, Client
Client(numwkrs)

for i in range(numwkrs):
  A=delayed(my_func)(i)
  Acombine.append(A)


Acombined.compute()
```

Outline

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

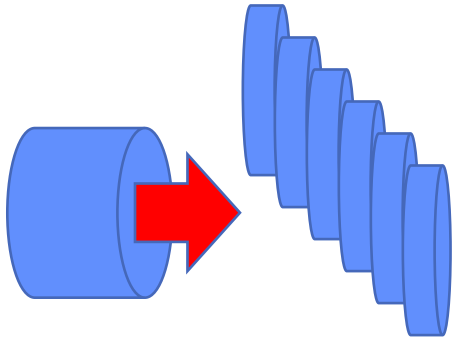
An option for (embarrassingly) Parallel R

1. Split up data into N parts



An option for (embarrassingly) Parallel R

1. Split up data into N parts



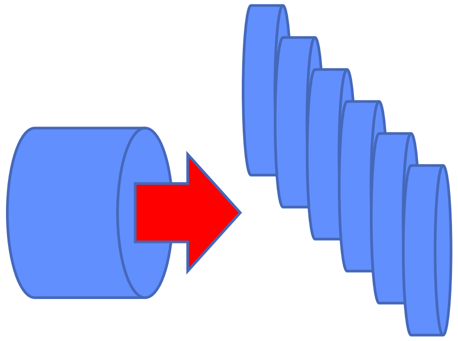
2. In slurm batch script:

mpirun …   my-perl-script

$$\begin{tikzcd}
\boxed{\begin{array}{l}\text{My-perl-script:}\\ \textit{get cpu-id \&}\\ \textit{pass it to R}\end{array}} \arrow[d] \\
\boxed{\begin{array}{l}\text{R script:}\\ \text{process}\\ \text{n-th dataset}\end{array}}
\end{tikzcd}$$

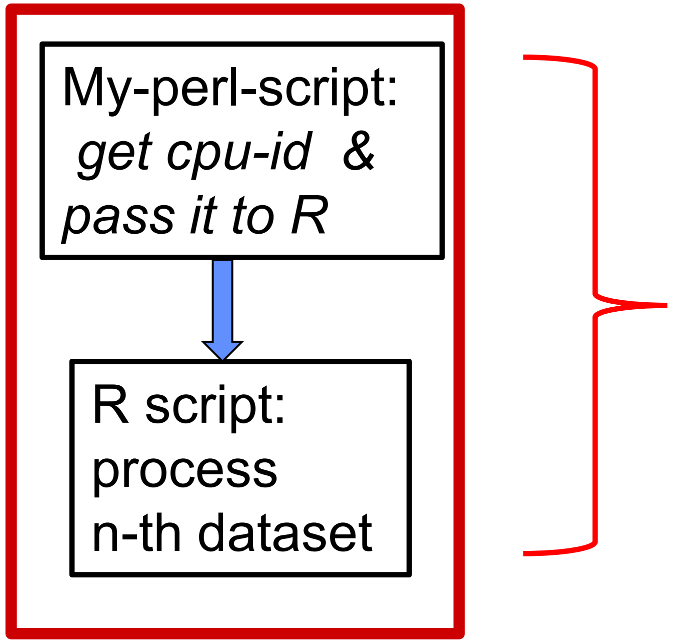
An option for (embarrassingly) Parallel R

1. Split up data into N parts



2. In slurm batch script:

mpirun …   my-perl-script



My-perl-script:
get cpu-id & pass it to R

R script:
process
n-th dataset

This gets distributed across nodes and cores by slurm & mpi parameters

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

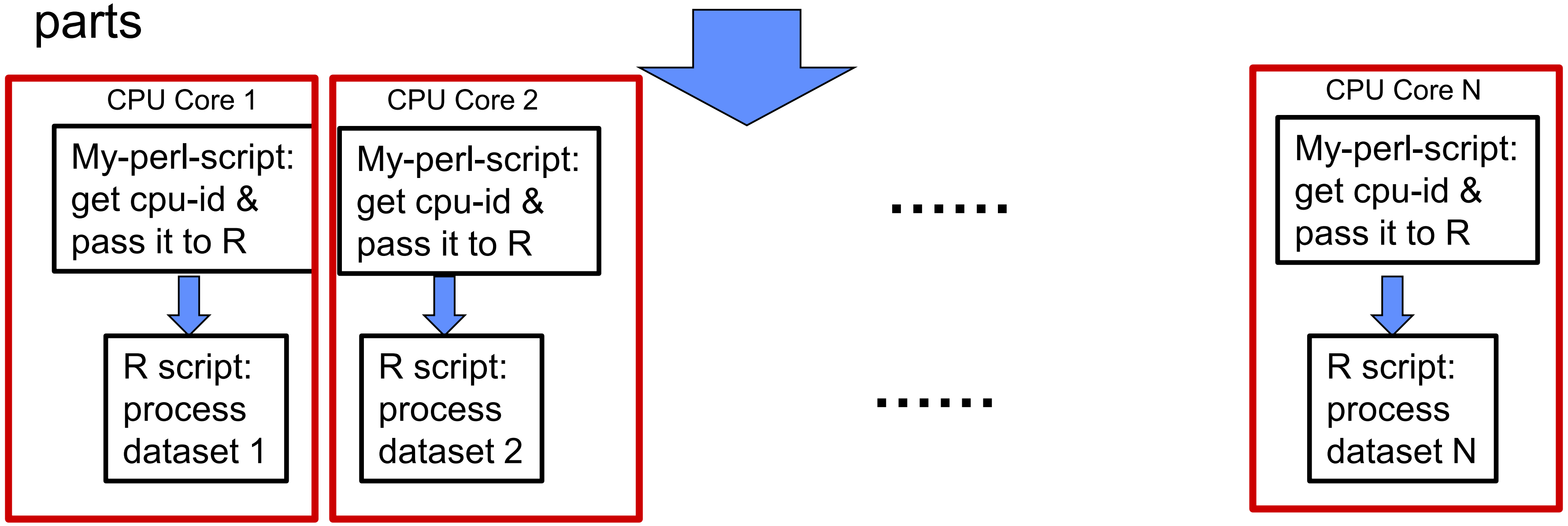
(based on /cm/shared/examples/sdsc/mpi-openmp-hybrid/hybrid-slurm.sb)

one R instance per core across all nodes

1. Split up data into N parts

2. In slurm batch script:

```
mpirun …    my-perl-script
```

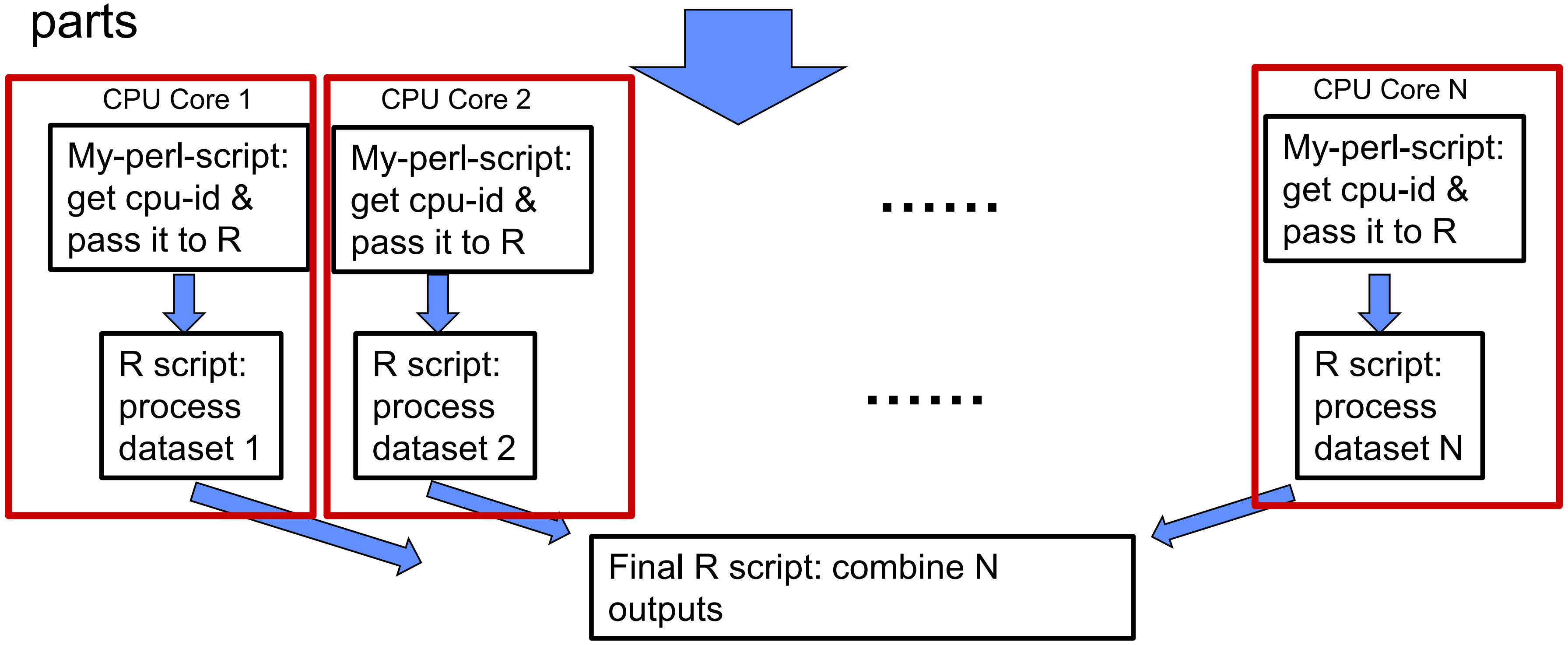

CPU Core 1
My-perl-script: get cpu-id & pass it to R
R script: process dataset 1
CPU Core 2
My-perl-script: get cpu-id & pass it to R
R script: process dataset 2
……
……
CPU Core N
My-perl-script: get cpu-id & pass it to R
R script: process dataset N

one R instance per core across all nodes

1. Split up data into N parts

2. In slurm batch script:

mpirun …   my-perl-script


CPU Core 1
My-perl-script: get cpu-id & pass it to R
R script: process dataset 1
CPU Core 2
My-perl-script: get cpu-id & pass it to R
R script: process dataset 2
……
……
CPU Core N
My-perl-script: get cpu-id & pass it to R
R script: process dataset N
Final R script: combine N outputs

More programming but perhaps more useful

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

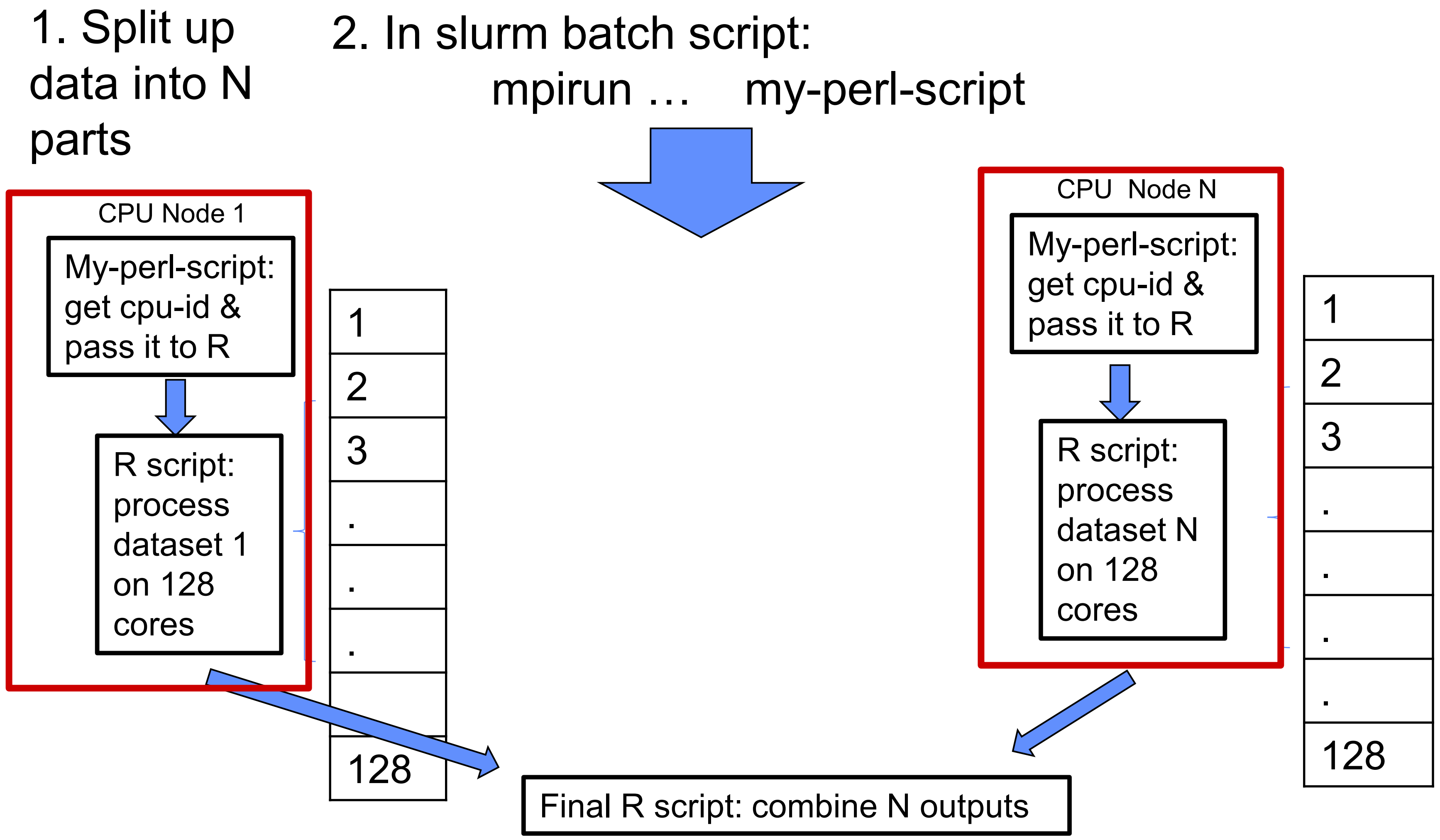
Normal batch job info

```
…
#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
CPU Node 1
My-perl-script: get cpu-id & pass it to R
R script: process dataset 1 on 128 cores
1
2
3
.
.
.
128
CPU Node N
My-perl-script: get cpu-id & pass it to R
R script: process dataset N on 128 cores
1
2
3
.
.
.
.
128
Final R script: combine N outputs

Outline

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

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** $L = \boldsymbol{MSE} + \boldsymbol{\lambda} \sum |b_{\boldsymbol{i}}|$

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

  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;

copies everything into binary file and 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

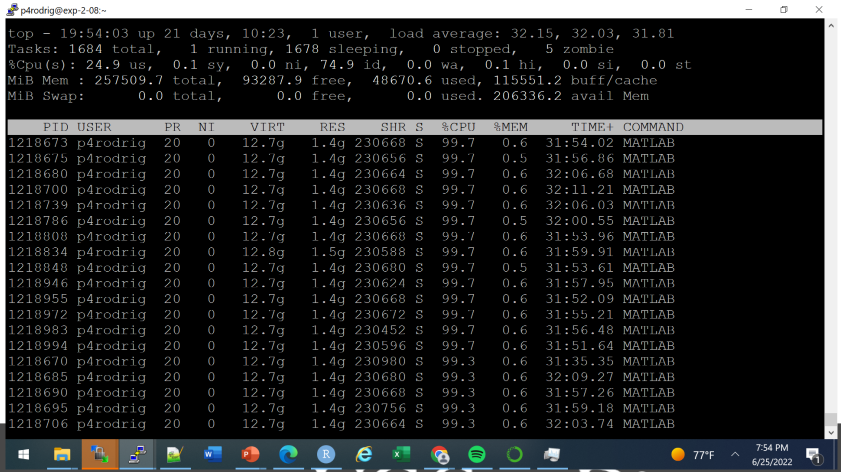
```
…..

#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 cpu
module load gcc/9.2.0
module load r

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 recalled a 'table2array' function that helped

But matlab kept failing with worker getting disconnected after 1 or 2 hours of processing; might be a memory issue



```
top - 19:54:03 up 21 days, 10:23,  1 user,  load average: 32.15, 32.03, 31.81
Tasks: 1684 total,   1 running, 1678 sleeping,   0 stopped,   5 zombie
%Cpu(s): 24.9 us,  0.1 sy,  0.0 ni, 74.9 id,  0.0 wa,  0.1 hi,  0.0 si,  0.0 st
MiB Mem : 257509.7 total,  93287.9 free,  48670.6 used, 115551.2 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 206336.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
1218673 p4rodrig  20   0   12.7g   1.4g 230668 S  99.7   0.6  31:54.02 MATLAB
1218675 p4rodrig  20   0   12.7g   1.4g 230656 S  99.7   0.5  31:56.86 MATLAB
1218680 p4rodrig  20   0   12.7g   1.4g 230664 S  99.7   0.6  32:06.68 MATLAB
1218700 p4rodrig  20   0   12.7g   1.4g 230668 S  99.7   0.6  32:11.21 MATLAB
1218739 p4rodrig  20   0   12.7g   1.4g 230636 S  99.7   0.6  32:06.03 MATLAB
1218786 p4rodrig  20   0   12.7g   1.4g 230656 S  99.7   0.5  32:00.55 MATLAB
1218808 p4rodrig  20   0   12.7g   1.4g 230668 S  99.7   0.6  31:53.96 MATLAB
1218834 p4rodrig  20   0   12.8g   1.5g 230588 S  99.7   0.6  31:59.91 MATLAB
1218848 p4rodrig  20   0   12.7g   1.4g 230680 S  99.7   0.5  31:53.61 MATLAB
1218946 p4rodrig  20   0   12.7g   1.4g 230624 S  99.7   0.6  31:57.95 MATLAB
1218955 p4rodrig  20   0   12.7g   1.4g 230668 S  99.7   0.6  31:52.09 MATLAB
1218972 p4rodrig  20   0   12.7g   1.4g 230672 S  99.7   0.6  31:55.21 MATLAB
1218983 p4rodrig  20   0   12.7g   1.4g 230452 S  99.7   0.6  31:56.48 MATLAB
1218994 p4rodrig  20   0   12.7g   1.4g 230596 S  99.7   0.6  31:51.64 MATLAB
1218670 p4rodrig  20   0   12.7g   1.4g 230980 S  99.3   0.6  31:35.35 MATLAB
1218685 p4rodrig  20   0   12.7g   1.4g 230680 S  99.3   0.6  32:09.27 MATLAB
1218690 p4rodrig  20   0   12.7g   1.4g 230668 S  99.3   0.6  31:57.26 MATLAB
1218695 p4rodrig  20   0   12.7g   1.4g 230756 S  99.3   0.6  31:59.18 MATLAB
1218706 p4rodrig  20   0   12.7g   1.4g 230664 S  99.3   0.6  32:03.74 MATLAB
```

Dask-ml

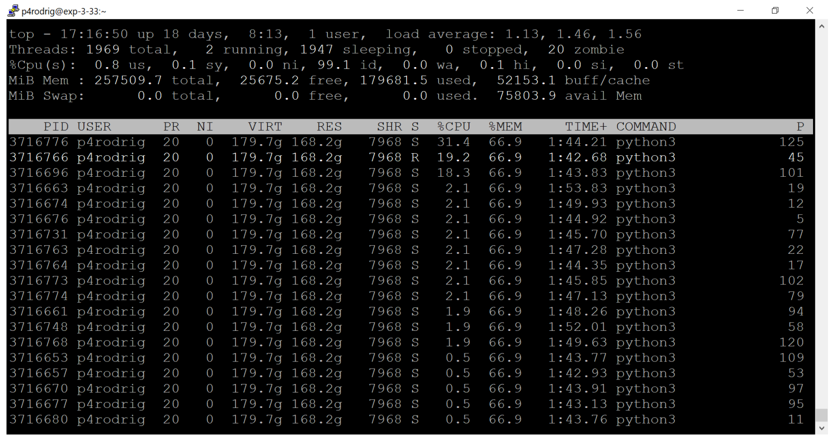
- Dask-ML – worked easily with distributed dataframe, built on top of sklearn,

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



```
top - 17:16:50 up 18 days,  8:13,  1 user,  load average: 1.13, 1.46, 1.56
Threads: 1969 total,   2 running, 1947 sleeping,   0 stopped,  20 zombie
%Cpu(s):  0.8 us,  0.1 sy,  0.0 ni, 99.1 id,  0.0 wa,  0.1 hi,  0.0 si,  0.0 st
MiB Mem : 257509.7 total,  25675.2 free, 179681.5 used,  52153.1 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.  75803.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND                   P
3716776 p4rodrig  20   0  179.7g 168.2g   7968 S  31.4  66.9   1:44.21 python3                 125
3716766 p4rodrig  20   0  179.7g 168.2g   7968 R  19.2  66.9   1:42.68 python3                  45
3716696 p4rodrig  20   0  179.7g 168.2g   7968 S  18.3  66.9   1:43.83 python3                 101
3716663 p4rodrig  20   0  179.7g 168.2g   7968 S   2.1  66.9   1:53.83 python3                  19
3716674 p4rodrig  20   0  179.7g 168.2g   7968 S   2.1  66.9   1:49.93 python3                  12
3716676 p4rodrig  20   0  179.7g 168.2g   7968 S   2.1  66.9   1:44.92 python3                   5
3716731 p4rodrig  20   0  179.7g 168.2g   7968 S   2.1  66.9   1:45.70 python3                  77
3716763 p4rodrig  20   0  179.7g 168.2g   7968 S   2.1  66.9   1:47.28 python3                  22
3716764 p4rodrig  20   0  179.7g 168.2g   7968 S   2.1  66.9   1:44.35 python3                  17
3716773 p4rodrig  20   0  179.7g 168.2g   7968 S   2.1  66.9   1:45.85 python3                 102
3716774 p4rodrig  20   0  179.7g 168.2g   7968 S   2.1  66.9   1:47.13 python3                  79
3716661 p4rodrig  20   0  179.7g 168.2g   7968 S   1.9  66.9   1:48.26 python3                  94
3716748 p4rodrig  20   0  179.7g 168.2g   7968 S   1.9  66.9   1:52.01 python3                  58
3716768 p4rodrig  20   0  179.7g 168.2g   7968 S   1.9  66.9   1:49.63 python3                 120
3716653 p4rodrig  20   0  179.7g 168.2g   7968 S   0.5  66.9   1:43.77 python3                 109
3716657 p4rodrig  20   0  179.7g 168.2g   7968 S   0.5  66.9   1:42.93 python3                  53
3716670 p4rodrig  20   0  179.7g 168.2g   7968 S   0.5  66.9   1:43.91 python3                  97
3716677 p4rodrig  20   0  179.7g 168.2g   7968 S   0.5  66.9   1:43.13 python3                  95
3716680 p4rodrig  20   0  179.7g 168.2g   7968 S   0.5  66.9   1:43.76 python3                  11
```

Spark

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

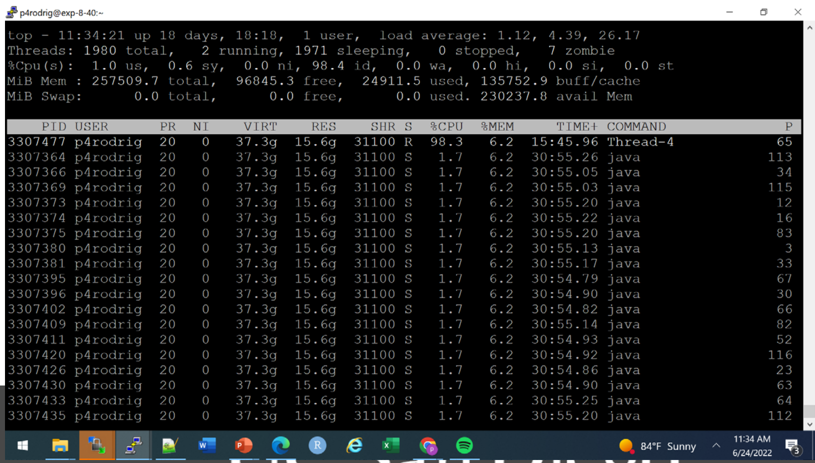
Use the vector assembler options to get Y and 'features' all in one dataframe

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



```
top - 11:34:21 up 18 days, 18:18,  1 user,  load average: 1.12, 4.39, 26.17
Threads: 1980 total,   2 running, 1971 sleeping,   0 stopped,   7 zombie
%Cpu(s):  1.0 us,  0.6 sy,  0.0 ni, 98.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 257509.7 total,  96845.3 free,  24911.5 used, 135752.9 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 230237.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND                       P
3307477 p4rodrig  20   0   37.3g  15.6g  31100 R  98.3   6.2  15:45.96 Thread-4                     65
3307364 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.26 java                        113
3307366 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.05 java                         34
3307369 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.03 java                        115
3307373 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.20 java                         12
3307374 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.22 java                         16
3307375 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.20 java                         83
3307380 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.13 java                          3
3307381 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.17 java                         33
3307395 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:54.79 java                         67
3307396 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:54.90 java                         30
3307402 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:54.82 java                         66
3307409 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.14 java                         82
3307411 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:54.93 java                         52
3307420 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:54.92 java                        116
3307426 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:54.86 java                         23
3307430 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:54.90 java                         63
3307433 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.25 java                         64
3307435 p4rodrig  20   0   37.3g  15.6g  31100 S   1.7   6.2  30:55.20 java                        112
```

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

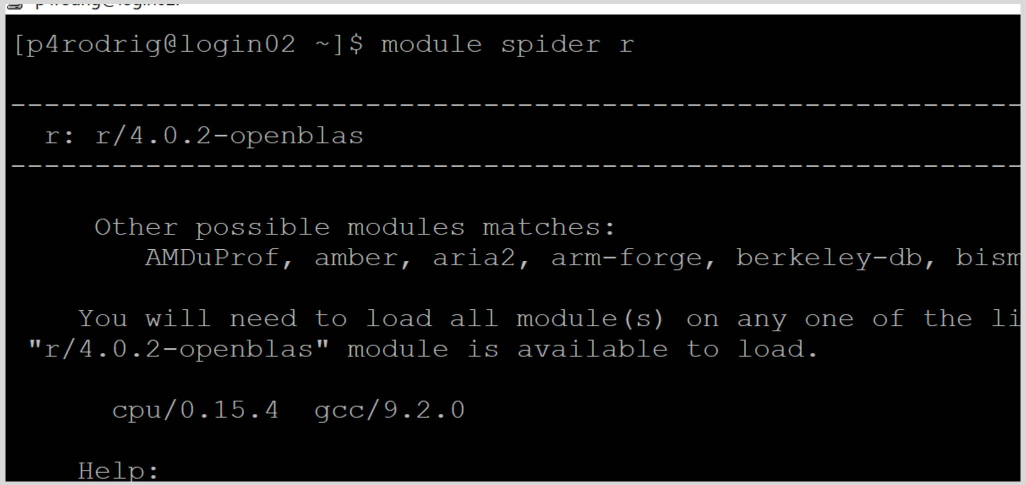
Some notes

- 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)*



```
[p4rodrig@login02 ~]$ module spider r

----------------------------------------------------------------------
  r: r/4.0.2-openblas
----------------------------------------------------------------------

     Other possible modules matches:
        AMDuProf, amber, aria2, arm-forge, berkeley-db, bism

    You will need to load all module(s) on any one of the li
 "r/4.0.2-openblas" module is available to load.

      cpu/0.15.4  gcc/9.2.0

    Help:
```

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