SDSC Summer Institute 2021
Scalable Machine Learning

Mai H. Nguyen & Paul Rodriguez

2021-August-04

Scalable Machine Learning Agenda

8:00 - 8:20 -- Machine Learning Overview

8:20 - 9:00 -- R on HPC

9:00 - 9:15 -- Break

9:15 - 10:15 -- Spark

10:15 - 10:45 -- Spark Hands-On

Scalable Machine Learning Agenda

8:00 - 8:20 -- Machine Learning Overview

8:20 - 9:00 -- R on HPC

9:00 - 9:15 -- Break

9:15 - 10:15 -- Spark

10:15 - 10:45 -- Spark Hands-On

R on HPC

Paul Rodriguez, Ph.D.

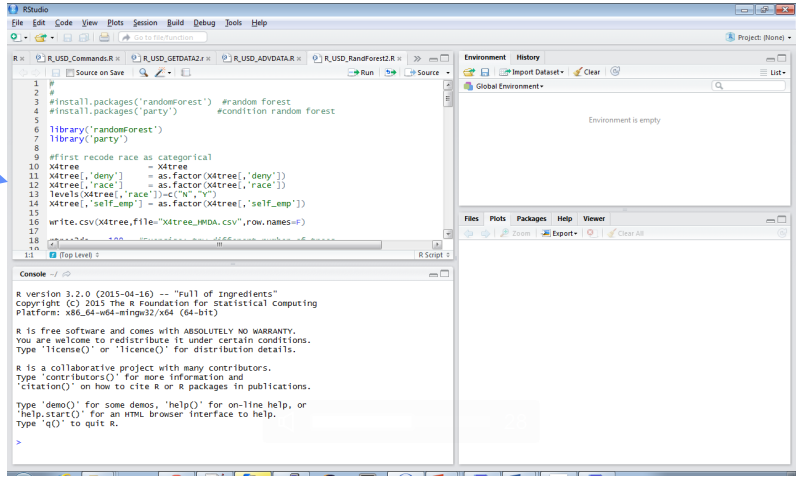
Table of Contents

A typical R development workflow

- R studio: An Integrated development environment for R

Environment Information on variables and command history

Edit window to Build scripts



R console

Plots, help docs, package lists

R commands in brief

- A typical R code workflow:

```
#READ DATA (housing mortage cases)
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
race              = as.numeric(X[,'s13'] %in% c(3,4)) #make race values 1-4 into values 0 or 1
deny              = as.numeric(X[,'s7']==3)            #make deny values into 0 or 1,
                                                            1 only for deny='3'
#RUN MODEL and SHOW RESULTS
lm_result         =lm(deny~race+pi_rat)          #lm is 'linearmodel'
summary(lm_result)
```

R strengths for HPC

- Data Wrangling

R strengths for HPC

- Data Wrangling
- Sampling/bootstrap methods

R strengths for HPC

- **Data Wrangling**
- **Sampling/bootstrap methods**
- **Particular Statistical procedures that you won't find implemented anywhere else, e.g.**

  Multiple Imputation methods,

  Instrument Variable (2 stage) Regression

  Matching subjects for pairwise analysis

  MCMC routines

Scaling, practically

- **Scaling (with or without more data):**
  - more complex analysis (ie optimizations)
  - more sampling (ie more trees in Random Forest)

Scaling, practically

- **Scaling (with or without more data):**
  - more complex analysis (ie optimizations)
  - more sampling (ie more trees in Random Forest)

- **Sometimes easy to parallelize (like with sampling)**

Scaling, practically

- **Scaling (with or without more data):**
  - more complex analysis (ie optimizations)
  - more sampling (ie more trees in Random Forest)

- **Sometimes easy to parallelize (like with sampling)**

- **Sometimes maybe too much communication between parts (like with matrix inversion)**

R Scaling In a nutshell

- R takes advantage of math libraries (like BLAS) for vector and linear algebra operations

```
> 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
```

R Scaling In a nutshell

- R takes advantage of math libraries (like BLAS) for vector and linear algebra operations
- R packages provide multicore, multinode, or distributed data (SparkR) options

R Scaling In a nutshell

- **R takes advantage of math libraries (like BLAS) for vector and linear algebra operations**
- **R packages provide multicore, multinode, or distributed data (SparkR) options**
- **However, model implementations are not necessarily built to use parallel backends**
  - Some models more amenable to parallel versions

Consider Regression Computations

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

  where Y=outcomes , X=data matrix

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)

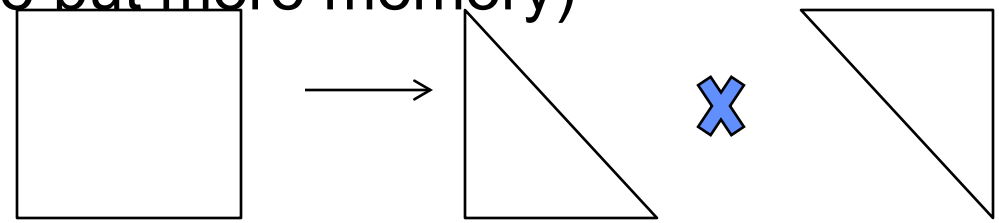
Consider Regression Computations

- **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)



Consider Regression models in R

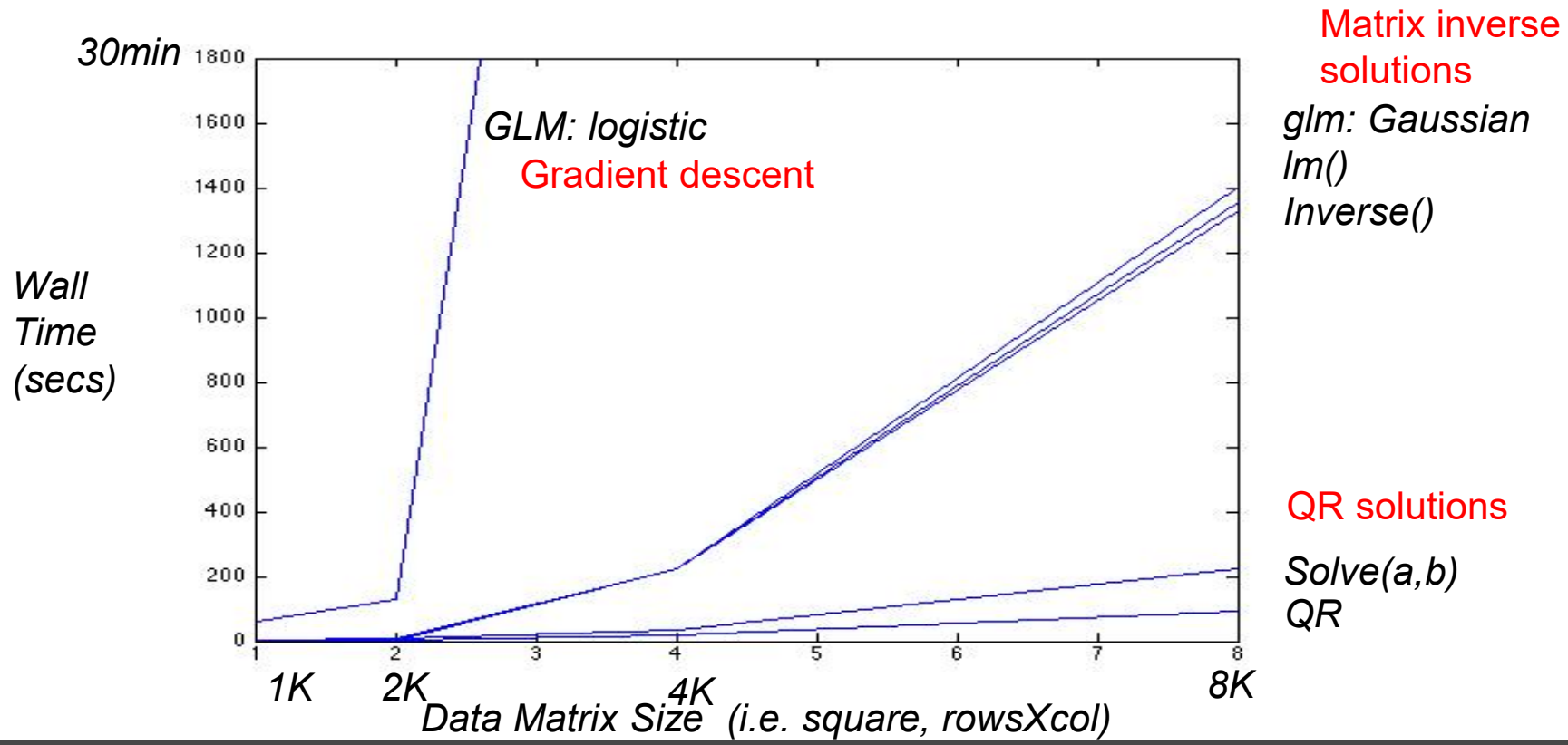
- **Related Models and Functions :**

lm() Linear Model

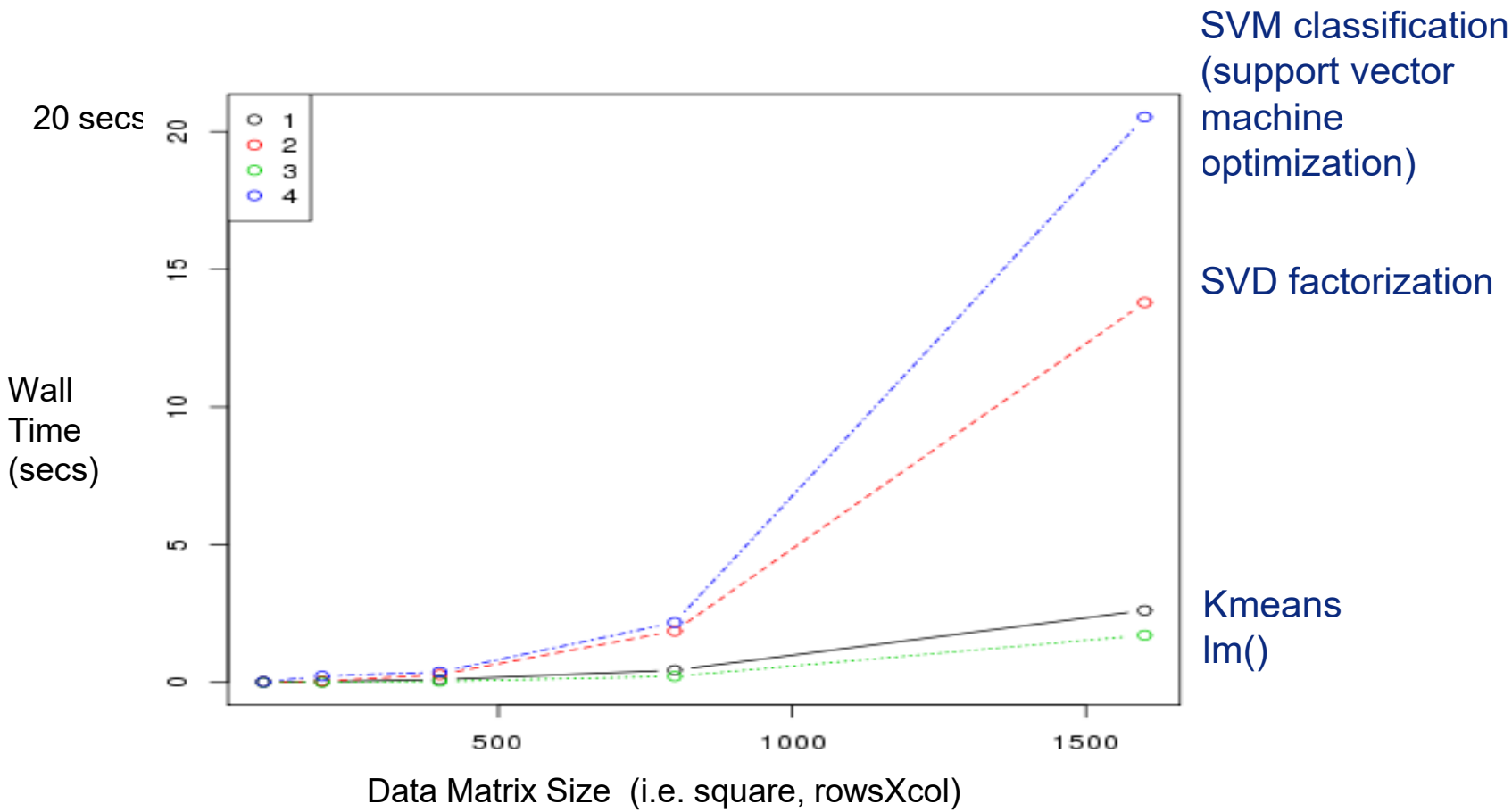
glm() Generalized Linear Model
(logistic regression,
LASSO version from Hastie et al.,etc)

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)



Machine learning models: Performance on 1 compute node



R multicore processing

- ‘doParallel’ package – provides the back end to the ‘for each’ parallel processing command

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

R multicore processing

- ‘doParallel’ package – provides the back end to the ‘for each’ parallel processing command
- uses threads across cpu cores, or Rmpi across nodes, to pass data & commands
- Updates and combines older packages

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 it thinks is need to every core

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

```
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)
```

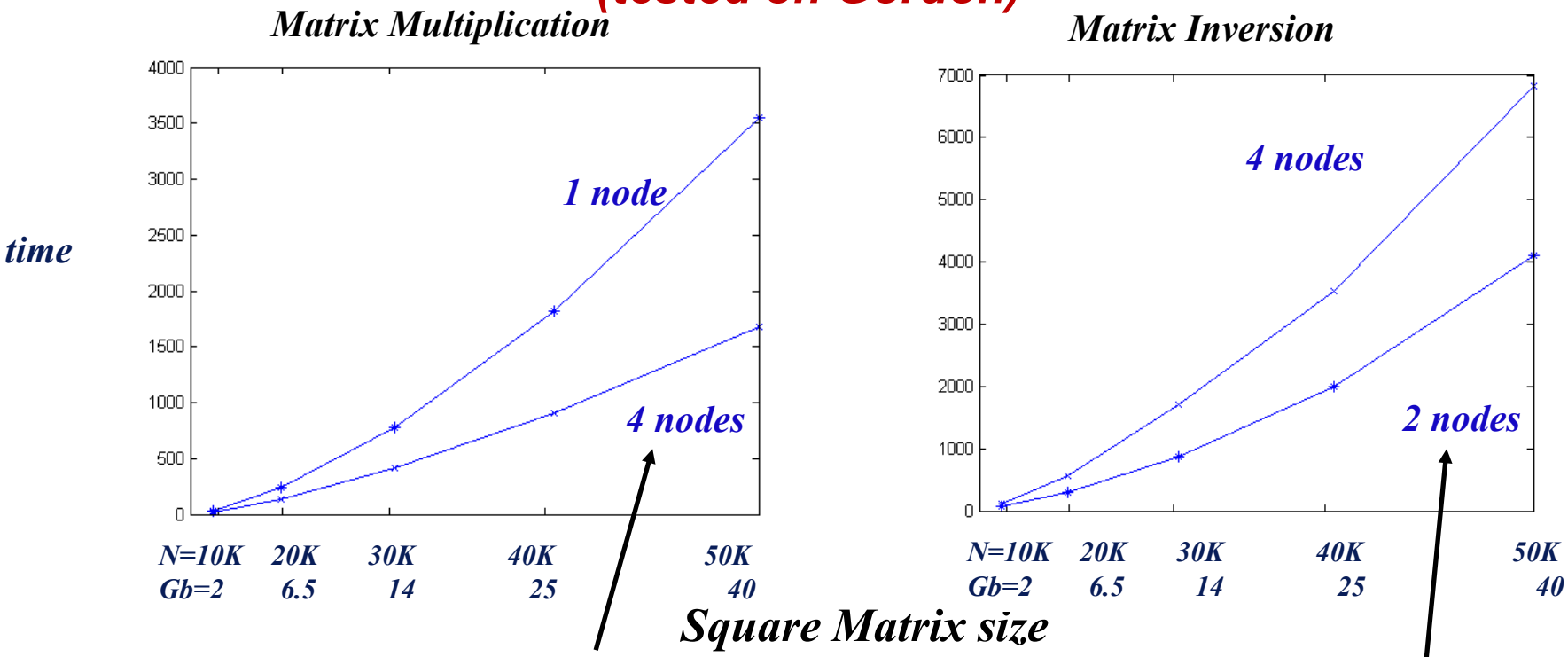
1. allocate cluster as parallel backend

2. %dopar% puts loops across cores and nodes

BEWARE: foreach will copy data it thinks is need to every core and node

Multiple Compute Nodes not always help

(tested on Gordon)

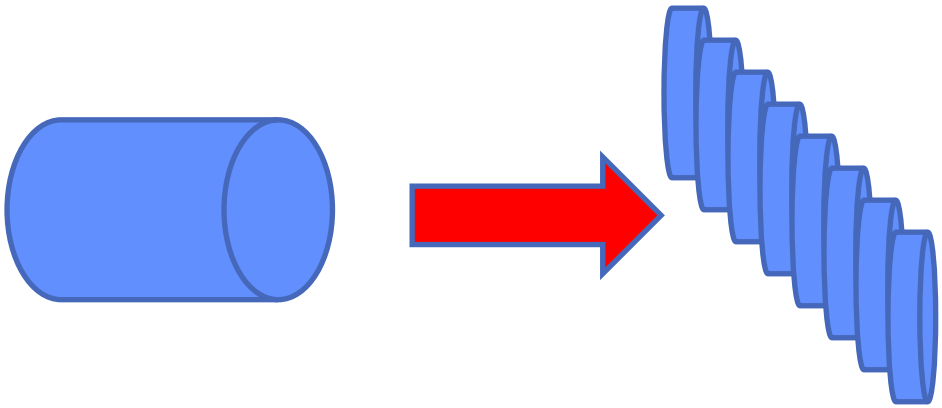


multinodes: more nodes is less time for multiplication,

less nodes is better for inversion

An option for (embarrassingly) Parallel R

1. Split up data into N parts

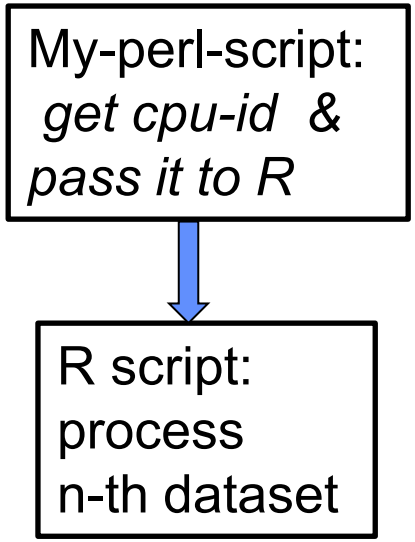


An option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:

mpirun … my-perl-script

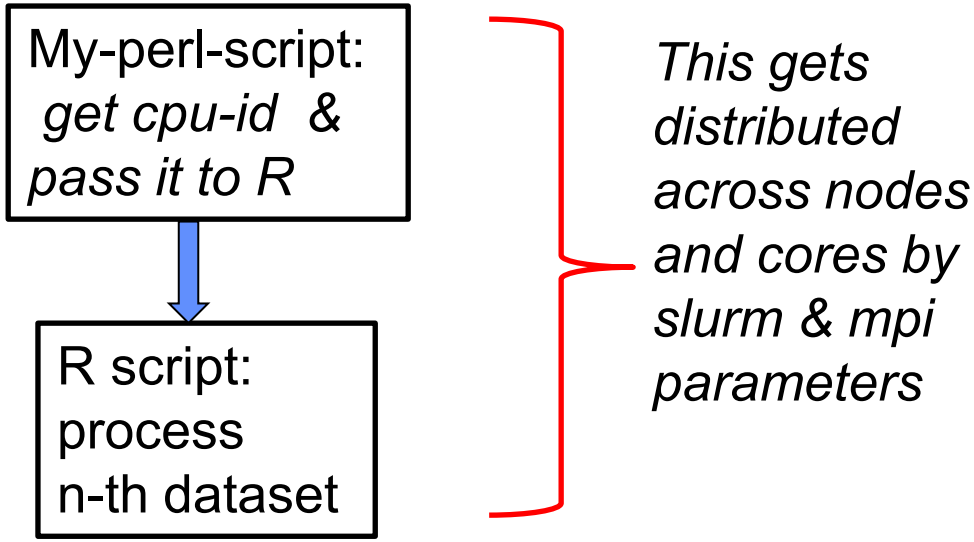


An option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:

mpirun … my-perl-script



An option for (embarrassingly) Parallel R

1. Split up data into N parts

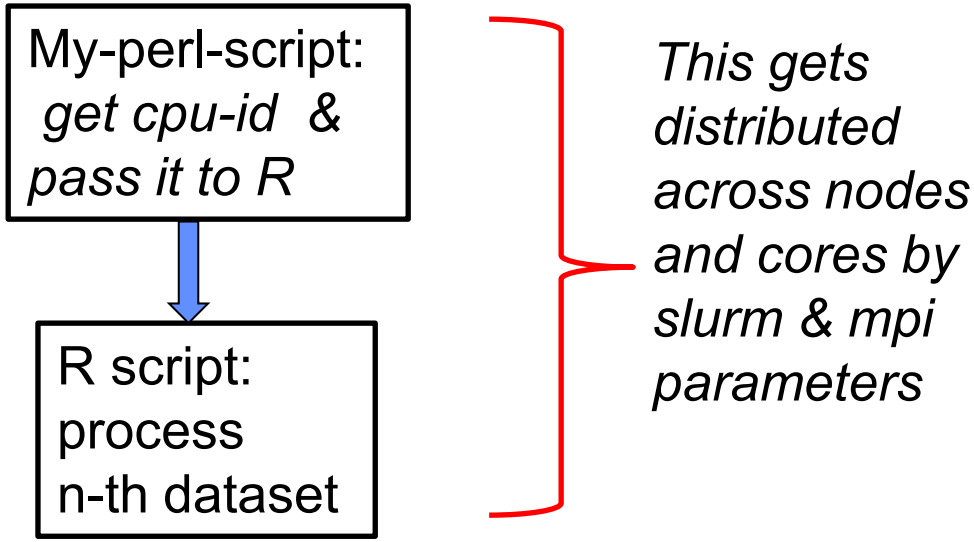
2. In slurm batch script:

mpirun … my-perl-script

Two Ways to run:

one R instance per core across all nodes

one R instance per node with 128 cores per R instance



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

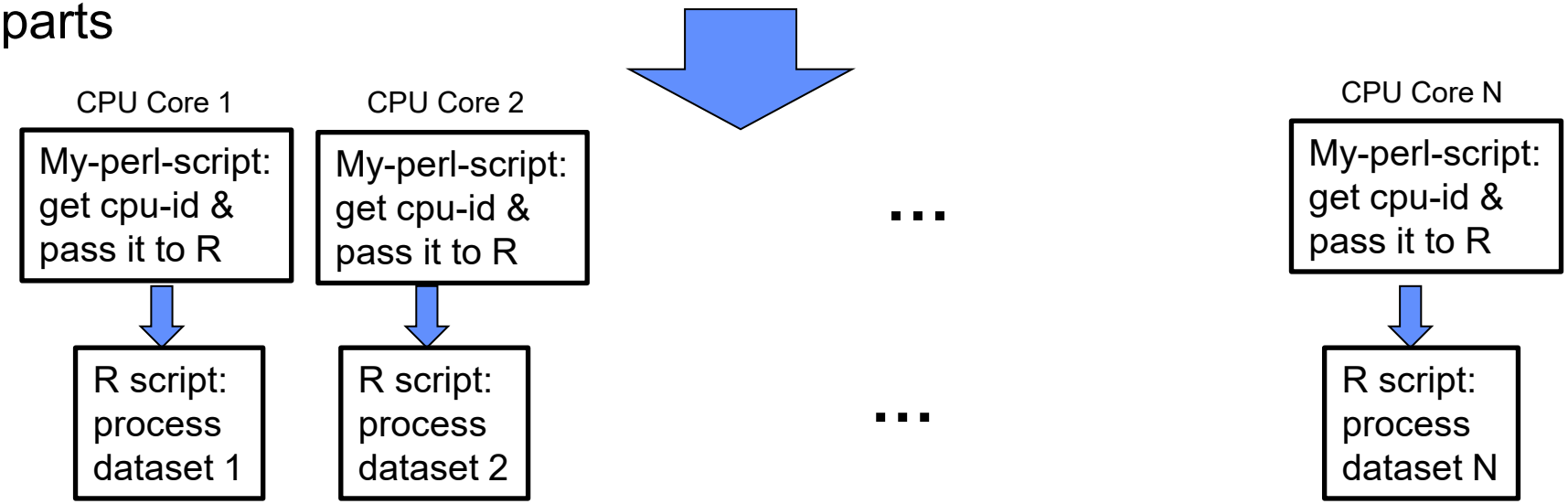
(based on /cm/shared/examples/sdsc/mpi-openmp-hybrid/hybrid-slurm.sb)

one R instance per core across all nodes

1. Split up data into N parts

2. In slurm batch script:

mpirun … my-perl-script

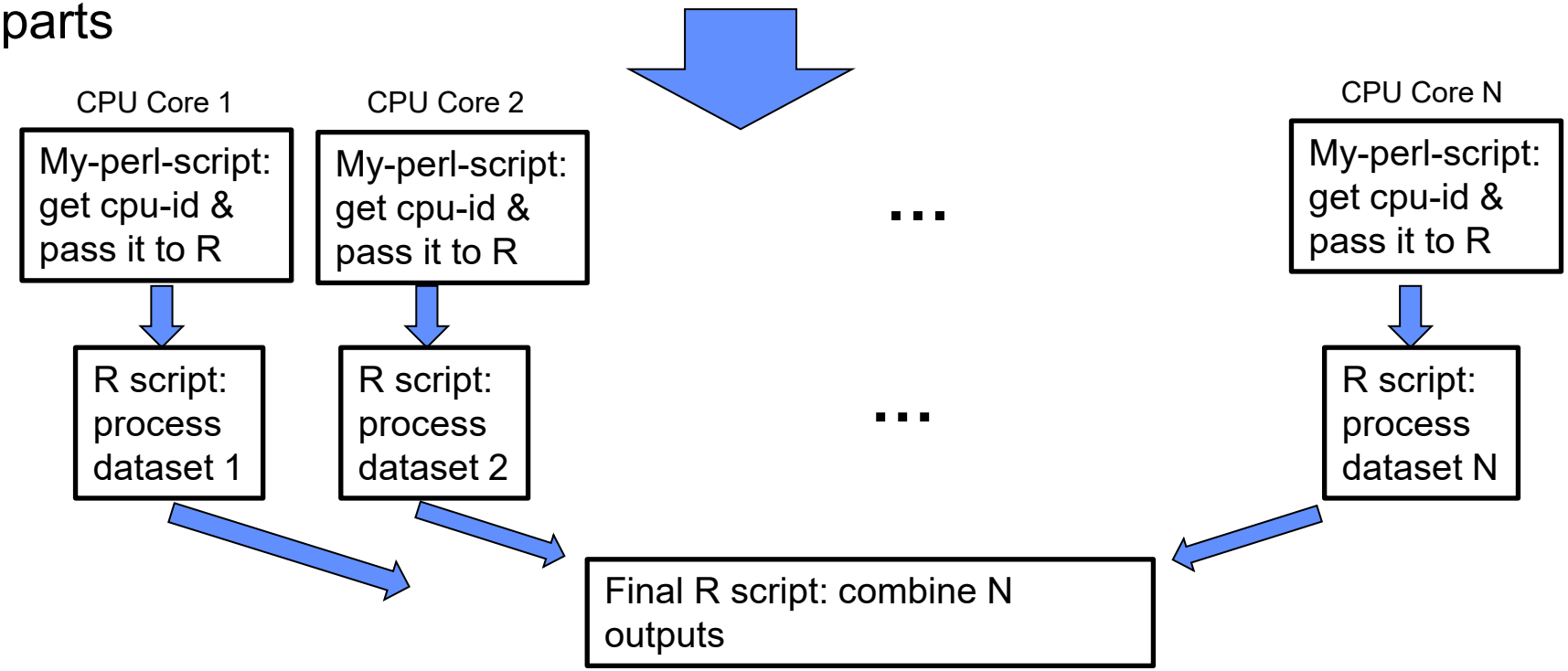


one R instance per core across all nodes

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

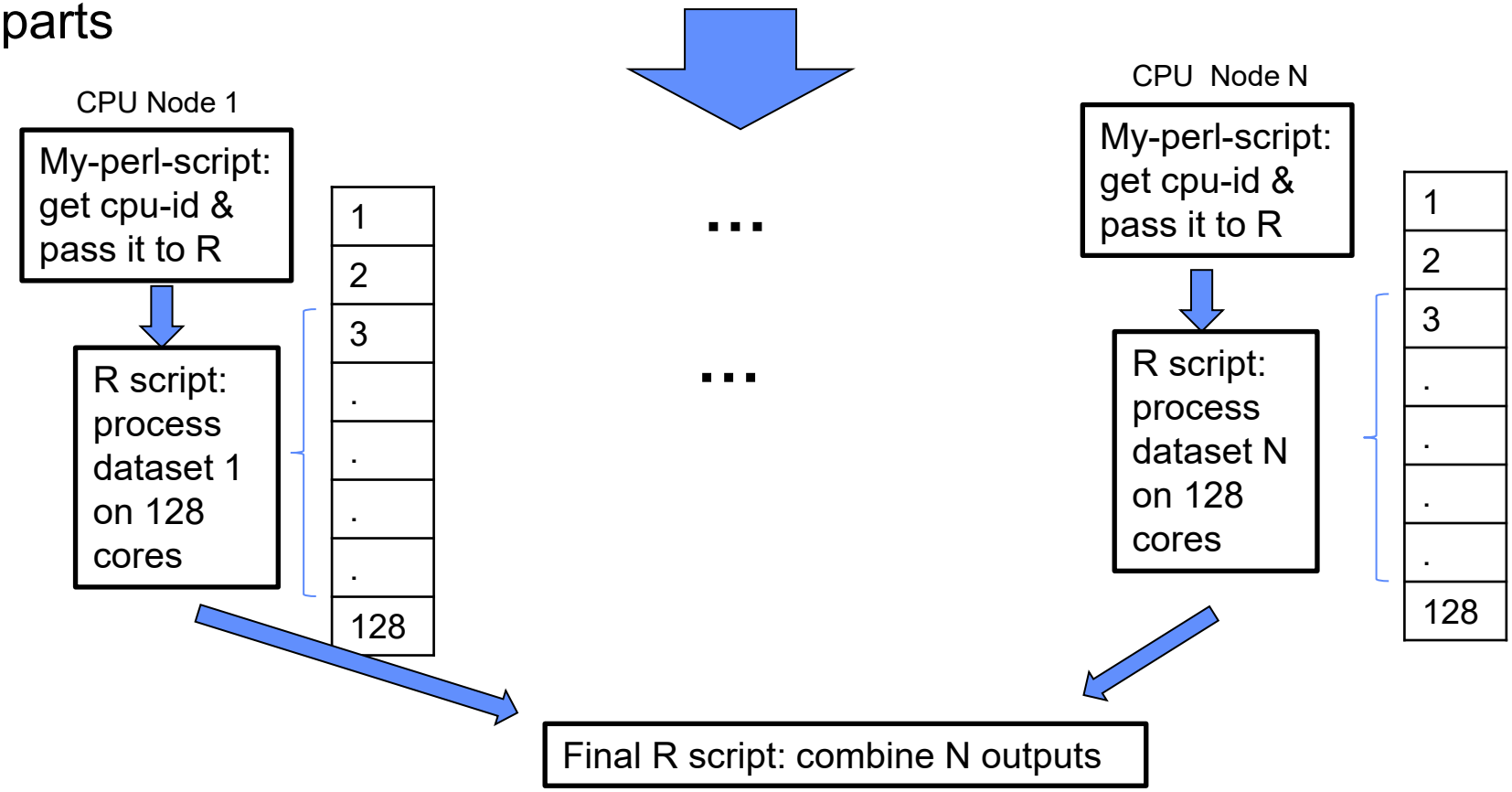
2 perl script/R instances
128 cores each
(doParallel can use them)

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



Example: scaling MCMC

Distributed Markov Chain Monte Carlo for Bayesian Hierarchical Models, Frederico Bumbaca, UCIrvine,

- *MCMC calculates the joint probability P(Data, Parameters) 1000s of times in a sequence (must be serial)*
- *Sometimes parameters can be 'partitioned'*
- *Using R package, "bayesm", it was run on Comet with embarrasing parallelization*

# Individuals	Cores	Individ per Core	Total Minutes (I/O time)
100 million	**1,7282 (max)**	**~ 58K**	**206 (38)**

Another MCMC option: “Stan”

R or Python interface, with many options

you write a script → Stan translates into C++ code

script sets up probability:

$$P(data, parameters) = \prod P(data_i, parameters)$$

Another MCMC option: “Stan”

R or Python interface, with many options

you write a script → Stan translates into C++ code

script sets up probability:

$$P(data, parameters) = \prod P(data_i, parameters)$$

If you set up log likelihood:

$$\log P(data, parameters) = \sum_i^N \log P(data_i, parameters)$$

Another MCMC option: “Stan”

R or Python interface, with many options

you write a script → Stan translates into C++ code

script sets up probability:

$$P(data, parameters) = \prod P(data_i, parameters)$$

If you set up log likelihood:

$$\log P(data, parameters) = \sum_i^N \log P(data_i, parameters)$$

then Stan will partition the data across cores.

Other R packages:

Also, for big data or big matrix

- Rspark - R interface to Spark (upcoming session)
- pdbR - distributed matrix support (better for dense matrices vs Spark)

Also:

- R openMP, Rmpi –
- Ff, bigmemory – map data to files
- Rgputools – GPU support

A note on using R in terminal window

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:**

  install.packages('package-name')

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

- **Sometimes on Comet, 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

- **pause**

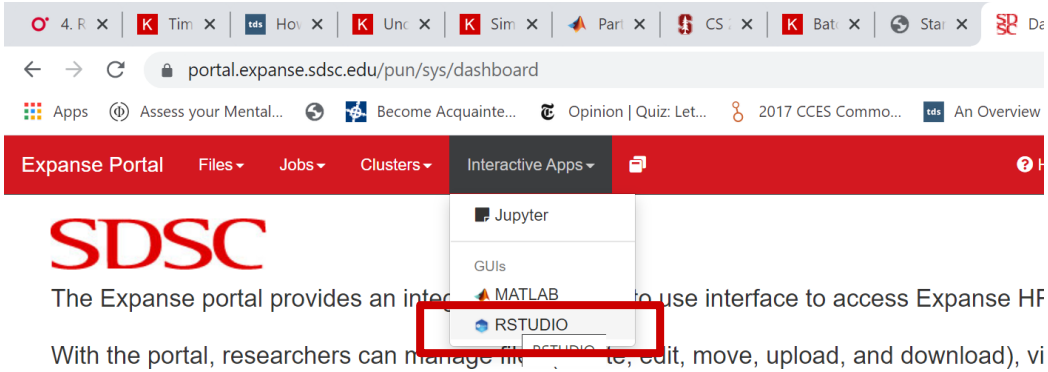
TestdoParallel R script

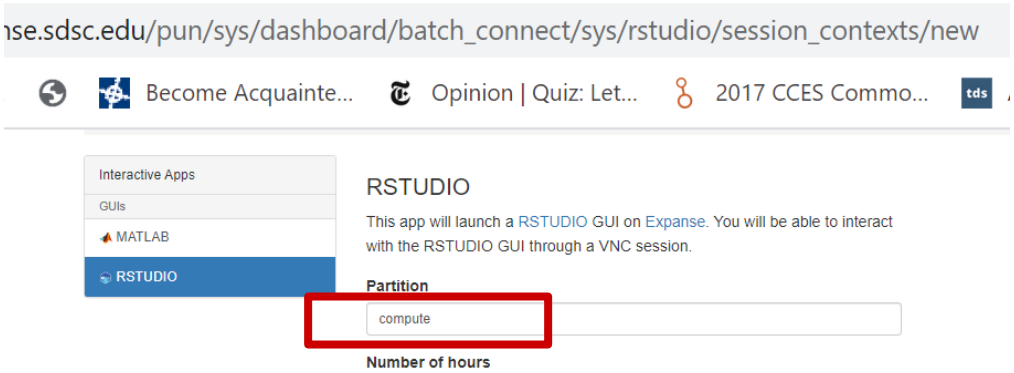
1 start Rstudio from portal (use shared and say <48 cores)
and run script
(it repeatedly does a regression)

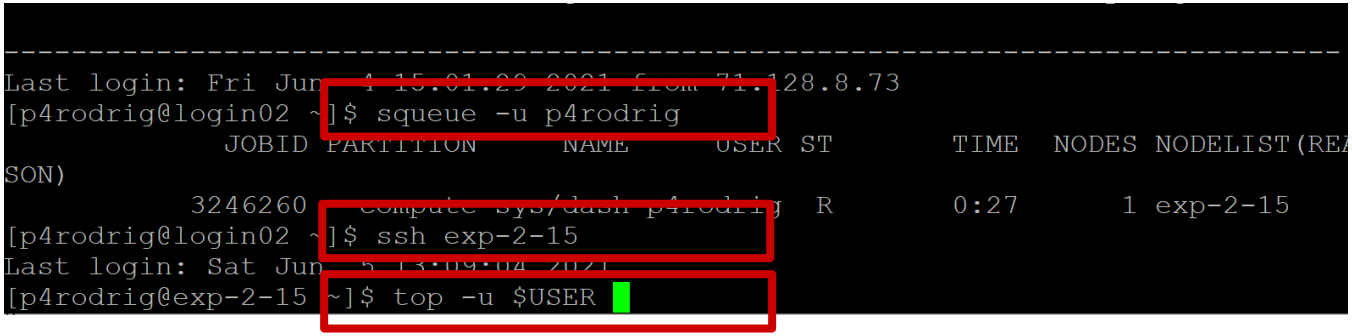
2 review execution using ‘top’ utility

3 vary the NxP matrix size or number of cores

Goal: look for tradeoffs in memory vs execution
e.g. If N gets too large then use less cores







Open portal ->
Interactive Apps ->
Rstudio

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

Also login:
login.expanse

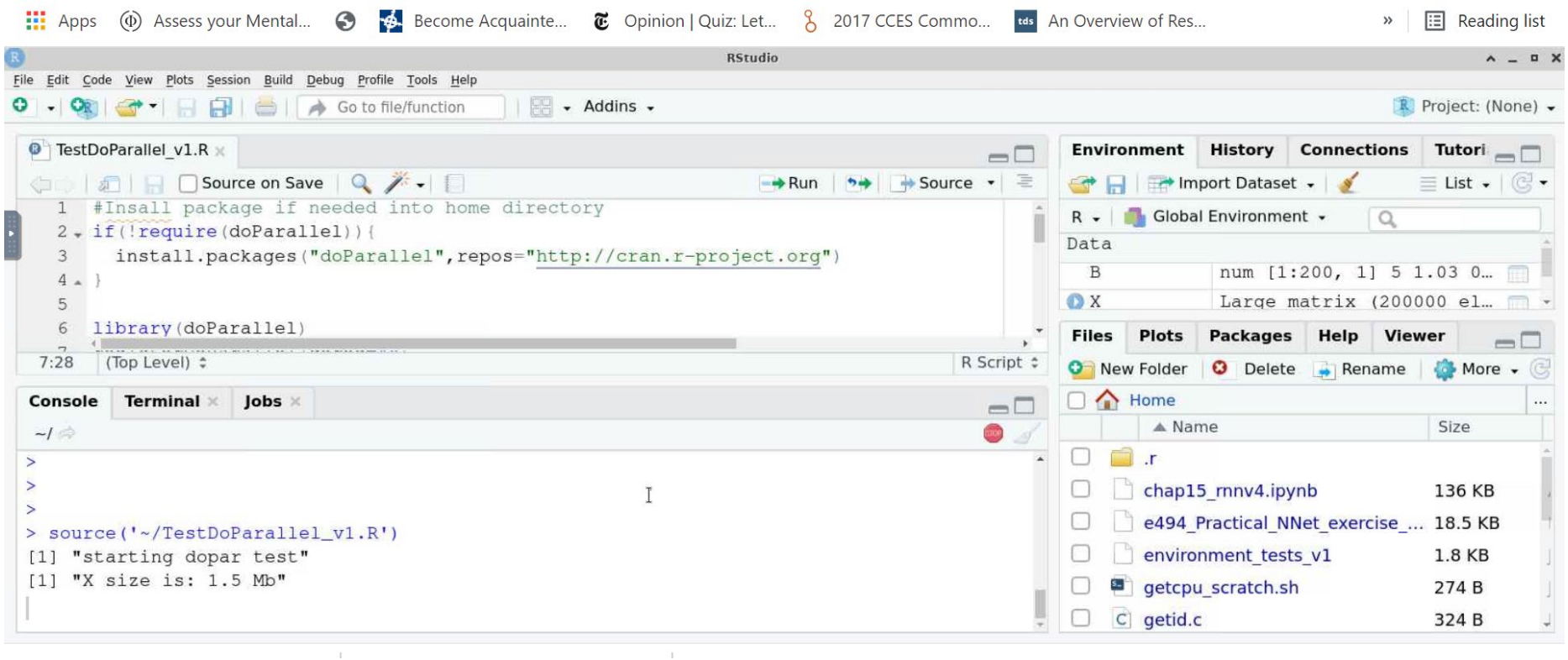
$ squeue –u $USER
$ ssh exp-##-##
$ top –u $USER

'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



Review the top output

Notice the elapsed time and memory size

Change the NxP matrix size and rerun

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

p4rodrig@exp-9-27:~

```
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
```

7:28 (Top Level)

Console Terminal Jobs

~/

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

2021_CIMLapplica....xlsx downloa

Source on Save

```
# 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
```

16:28 (Top Level)

Console Terminal Jobs

~/

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

2021_CIMLapplica....xlsx d

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?

THE END