OAK RIDGE National Laboratory

Statistical Computing on Large Parallel Architectures

George Ostrouchov[1,2,3] and Drew Schmidt[2]

[1]Scientific Data Group, CSMD

[2]Advanced Data and Workflow Group, NCCS

Oak Ridge National Laboratory

[3]Business Analytics and Statistics, UTK (Joint Faculty)

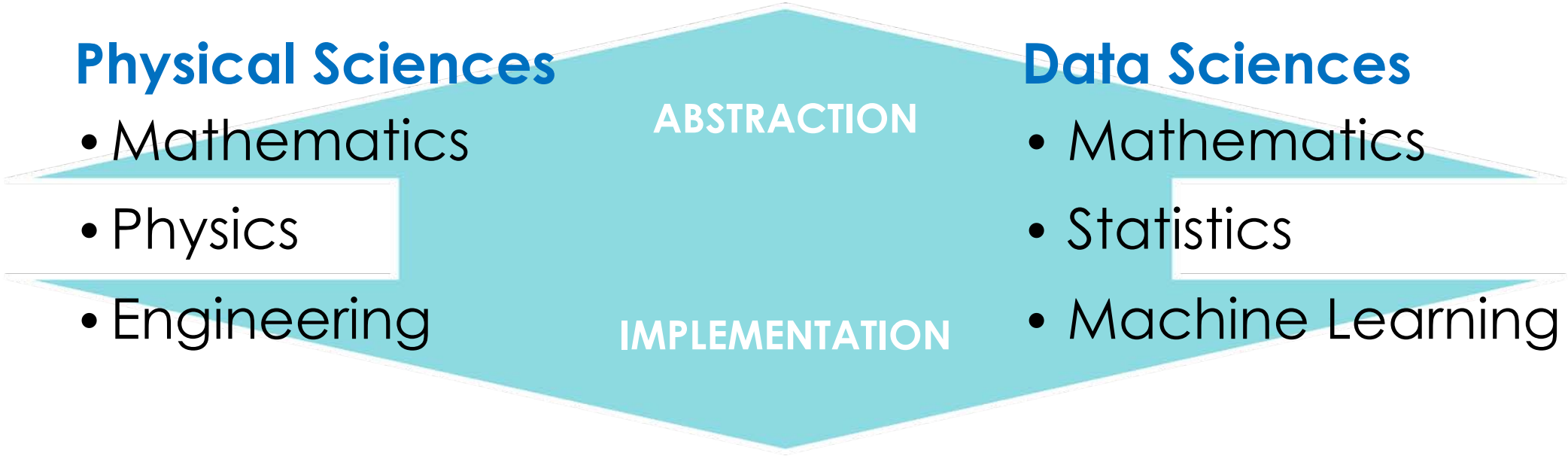
University of Tennessee, Knoxville

Advanced Statistics meets Machine Learning III Workshop, Argonne National Lab, November 13-15, 2019

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Statistics is to Machine Learning as Physics is to Engineering

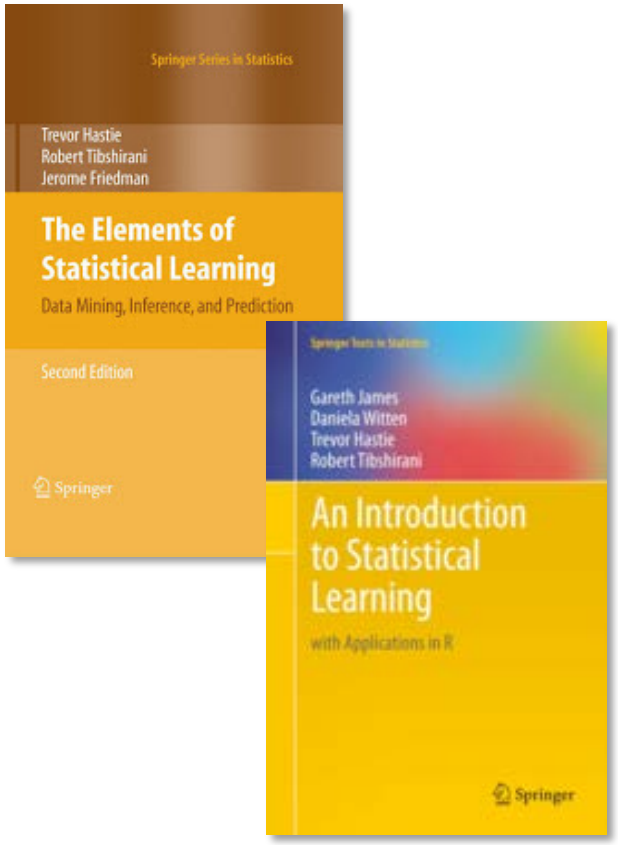


Statistics is the "Physics" of Data

Machine Learning or Statistical Learning?

Both are about **learning from data**, some difference in emphasis:

- SL: Explainability and discovery, uncertainty is fundamental
- ML: Processing and prediction, uncertainty is optional

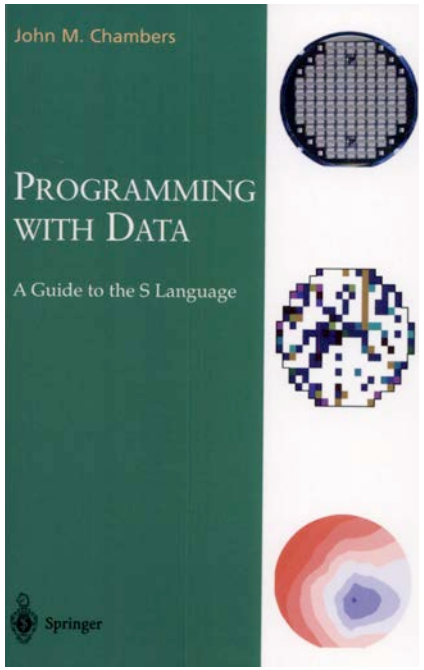


Dictionary

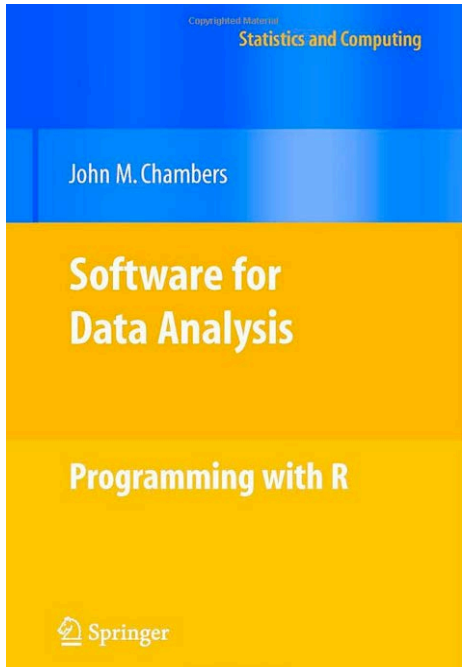
Machine Learning	Statistical Learning
Accuracy	(lack of) Bias
Bias	Intercept
Generalization	Test set performance or Expected prediction error
Inference	Prediction, point estimate
Learning	Fitting
Network, graph	Model
One-hot	Dummy variable
Precision	(lack of) Variability
Supervised learning	Regression/classification
Uncertainty	Inference
Unsupervised learning	Density estimation, clustering
Weights	Parameters

Adapted from Rob Tibshirani (Stanford) and expanded

Language for Programming with Data: R



1998 S language



2008 R language

Same language, different engine

2019 IEEE Spectrum's Ranking of Programming Languages

Rank	Language	Type	Score
1	Python		100.0
2	Java		96.3
3	C		94.4
4	C++		87.5
5	R		81.5
6	JavaScript		79.4
7	C#		74.5
8	Matlab		70.6
9	Swift		69.1
10	Go		68.0

Diversity for Data and Depth for Statistics in R

Bringing Modern Statistical Science and R to HPC

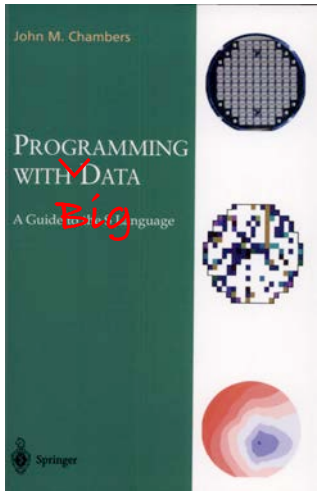
Core Team (leadership and lead developers)

Wei-Chen Chen, US FDA

George Ostrouchov, ORNL & UTK

Drew Schmidt, ORNL

pbdR
Programming with Big Data in R

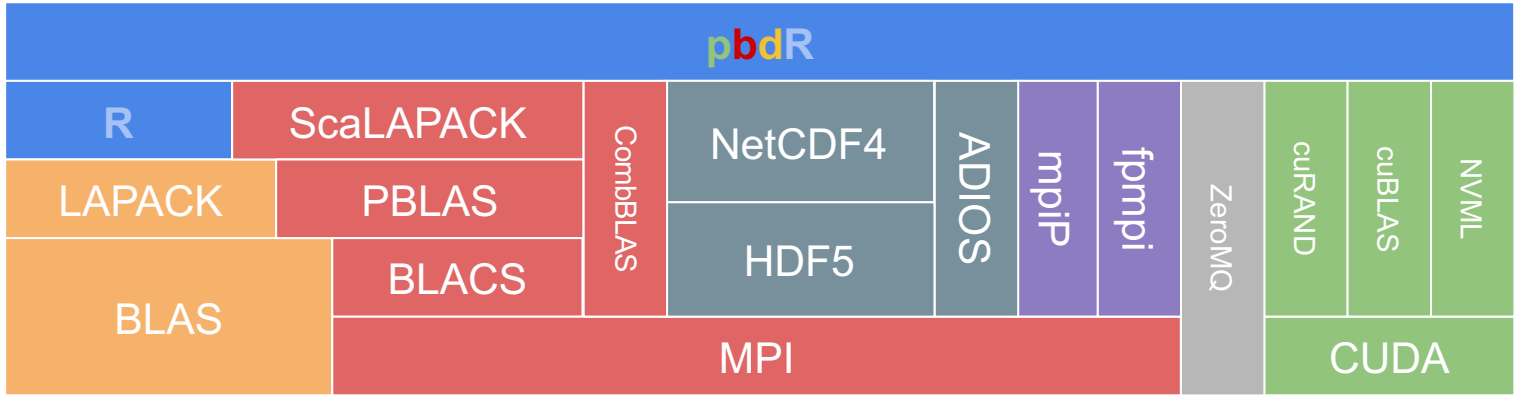


Developers

Christian Heckendorf, Yuping Lu, Michael Matheson, Pragneshkumar Patel, Gaurav Sehrawat , Amil Williamson

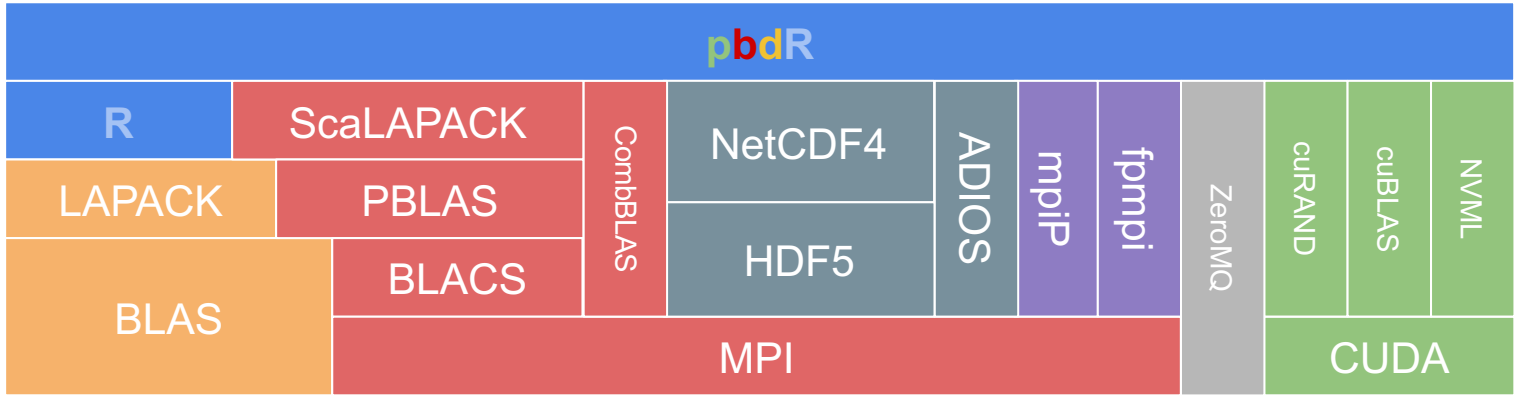
Contributors

Moustaffa Ahmed, Whit Armstrong, Ewan Higgs, Sébastien Lamy de la Chapelle, Michael Lawrence, Andrey Paul, A. Philipp, David Pierce, Brian Ripley, Elliott Sales, Aiello Spencer, ZhaoKang Wang, Hao Yu



pbdR Philosophy

- Bridge high-performance computing with high-productivity of R language
- Keep syntax *identical* to R, when possible.
- Software reuse philosophy:
  - Don't reinvent the wheel when possible
  - Introduce HPC standards with R flavor
  - Use scalable HPC libraries with R convenience
- Simplify and use R intelligence where possible



"… pbdR … outperformed all the other systems in almost all cases on dense data."

"Overall, pbdR is best suited to users who want to rapidly prototype new LA-based analysis algorithms at scale"

- An independent comparison on small distributed systems with up to 8 nodes (224 cores)
- Software considered: **pbdR , MADlib**, **MLlib**, **SystemML**, and **TensorFlow**
- Computations considered: distributed analytics based on linear algebra

Anthony Thomas, Arun Kumar: **A Comparative Evaluation of Systems for Scalable Linear Algebra-based Analytics**. *Proceedings of the VLDB Endowment,* Volume **11,** No. **13,** September 2018, p. 2168-2182.

pbdR
Programming with Big Data in R

Parallel I/O

- `hdfio`
  - High-level utilities for working with data frames in HDF5 and sharing between Python and R
  - See `rhdf5` and `hdf5r` packages for lower level utilities
  - https://code.ornl.gov/olcf-analytics/summit/r/blob/master/content/pbdR/hdfio.md
- `pbdIO`
  - So far mostly for parallel reads of collections of CSV files
  - Chunking, data frame rebalancing
- `pbdADIOS`
  - ADIOS 1.8, can stream with in-memory files, needs development for transition to ADIOS 2.0.
- `pbdNCDF4`
  - NetCDF4 file collective parallel reads and writes
- In situ capabilities
  - Communicator management, in-memory streaming file reads

Advanced CPU Analytics

- Packages: `pbdDMAT` (and friends), `kazaam`, `pbdML`, `pmclust`, `pbdXGB`
  - k-means and Model-Based Clustering
  - GLM's
  - Fisher's Linear Discriminant
  - Randomized SVD/PCA, Robust PCA
  - Gradient Boosting Machines and Random Forests via XGBoost
- Parallel dense matrix and sparse matrix infrastructure for custom distributed methods

Striving for interactive performance on Terabyte-sized data analysis

HPC wire

July 6, 2016

"**OLCF Researchers Scale R to Tackle Big Science Data Sets**" . . . "***for situations where one needs interactive near-real-time analysis, the pbdR approach is much better*** *[than Apache Spark–like frameworks].*" PCA of a 134 GB matrix: "*hours on . . . Apache Spark, . . . less than a minute using R.*"

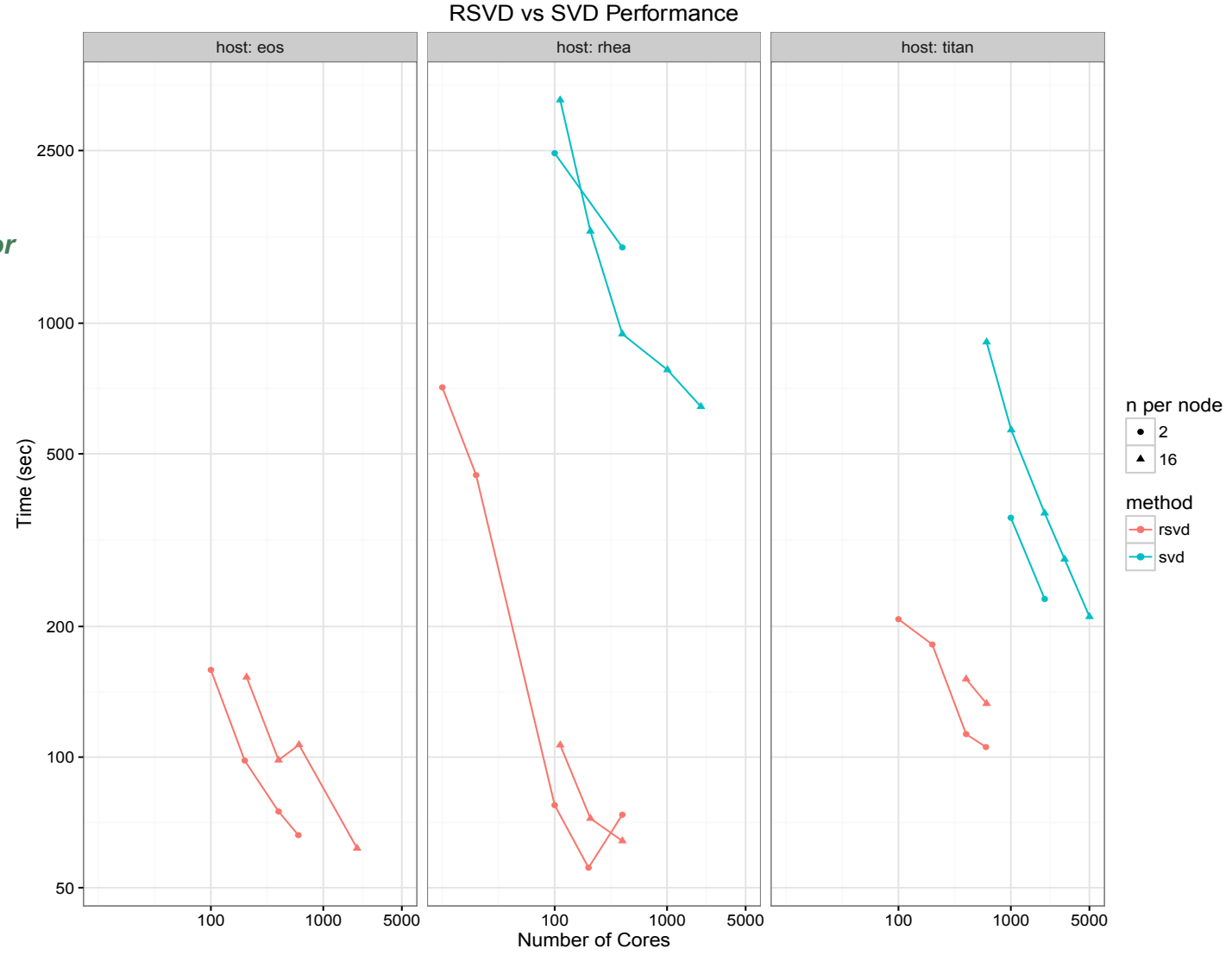
April 20, 2017

"**ORNL Researchers Bridge the Gap Between R, HPC Communities**" **. . . "untapped [R] domains" represent an enormous potential user base for world-class computers .**"

May 7, 2018

"ORNL Data Scientists Release pbdR 1.0"

pbdR
Programming with Big Data in R



OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

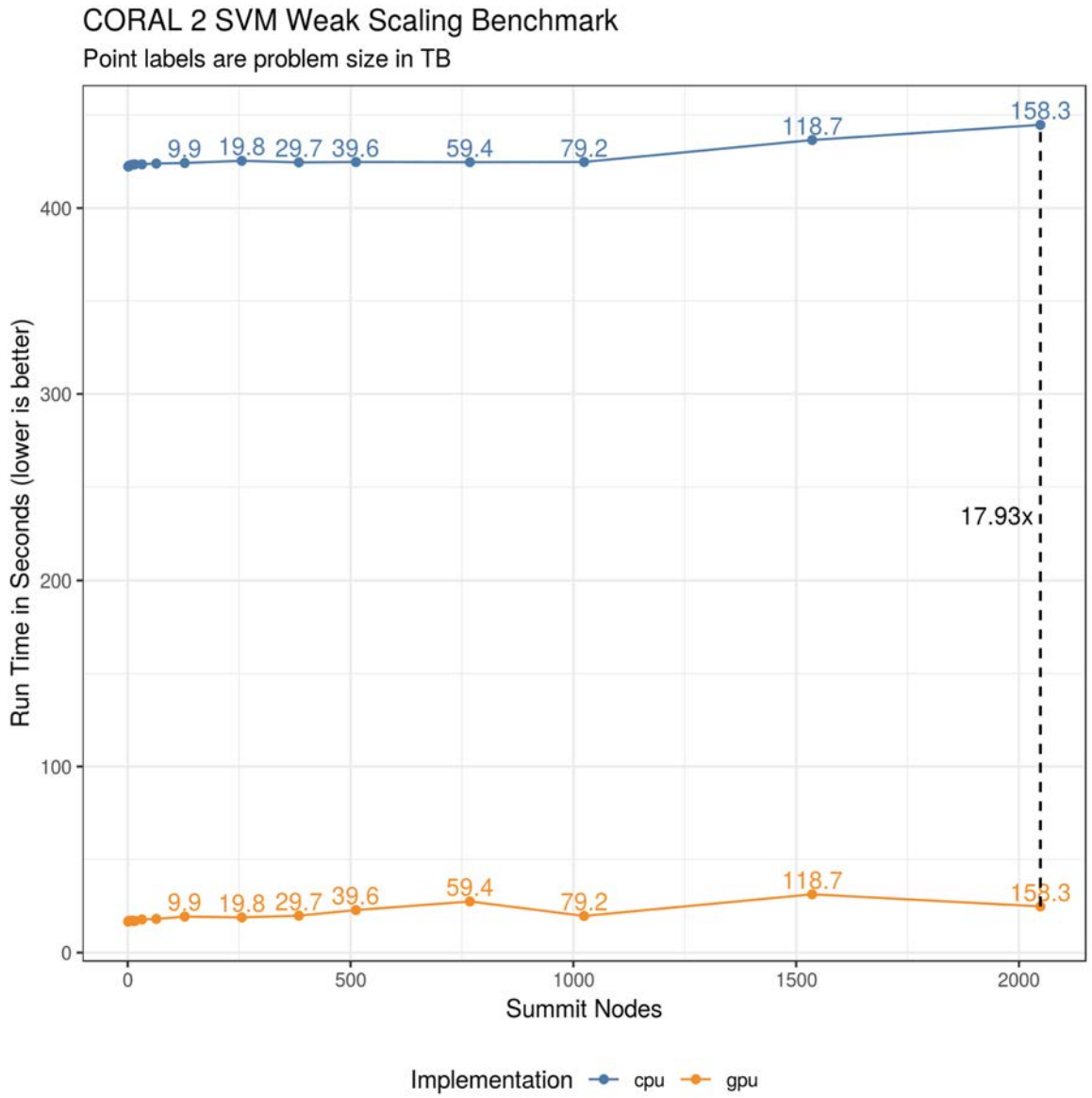
Seconds for 150 TB SVM Computation on Half of Summit

GPU Analytics benchmarks

- Packages: **`dimrgame, clustrgame, glmrgame`**
  - SVM, SVD, PCA, k-means

Example code with small toy data (benchmark data is more complex)

```
suppressMessages(library(kazaam))
suppressMessages(library(glmrgame))

data(iris)
is_setosa = expand((iris[, 5] == "setosa")*2 - 1)
iris = expand(as.matrix(iris[, -5]))
iris = cbind(shaq(1, nrow=nrow(iris), ncol=1), iris)

w_cpu = svm(iris, is_setosa)
w_gpu = svm_game(iris, is_setosa)

finalize()
```



Low-Level GPU Tools

nvsmi

```
options("nvsmi_printer"="minimal")
nvsmi::smi()
## +-----------------------------------------------------------------------------+
## | Date: Thu Apr 18 12:44:24 2019       Driver Version: 390.116                |
## |-------------------------------+----------------------+----------------------+
## | GPU Name                | Util  Fan  Temp   Perf        Power         Memory |
## |=========================+======================================================|
## |   0 GeForce GTX 107... |   0%  39%   32C     P2     42W/252W    584/8116MiB |
## +-------------------------------+----------------------+----------------------+
##
## +-----------------------------------------------------------------------------+
## |  GPU       PID   Type   Process name                                Mem Used |
## |=============================================================================|
## |    0      1781      G   /usr/lib/xorg/Xorg                           273MiB |
## |    0     21407      C   /usr/lib/R/bin/exec/R                        311MiB |
## +-----------------------------------------------------------------------------+
```

cur

```
library(cur)

a = 1:10
b = integer(10)

x = cudaMalloc(10, "int")
cudaMemcpy(x, a, 10, "int", "hosttodevice")
cudaMemset(x, 0, 4, "int")
cudaMemcpy(b, x, 10, "int", "devicetohost")

b
## [1]  0  0  0  0  5  6  7  8  9 10
```

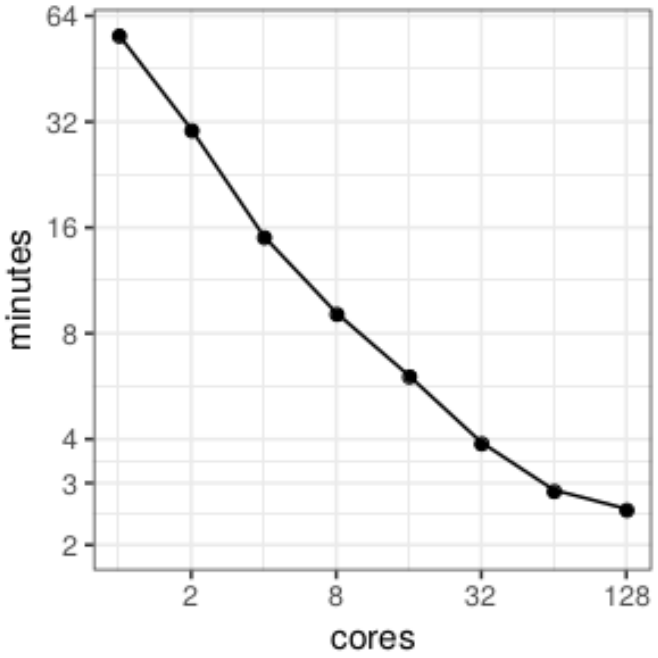
curand

```
n = 2.5e9
memuse::howbig(n)
## 18.626 GiB

system.time(stats::runif(n))
##  user  system elapsed
## 60.548   6.676  67.201
system.time(curand::runif(n))
##  user  system elapsed
## 4.792   7.952  12.739
```

HOSVD* Workflow for a Fusion XGC1 Simulation Output

- Tensor dimensions:
  - 41 (time)
  - 32 (toroidal angle)
  - 232,011 (unstructured poloidal mesh)
  - Total size ~ 2.3 GB
- Workflow computations:
  - Read data from 41 HDF5 files
  - 5 SVD computations in series
  - 6 tensor unfoldings
  - 80 pdf plots (~3 MB each)
- Powered by R and pbdR packages
  - `pbdMPI`, `kazaam`, `launchr`

Strong scaling for complete workflow



- Not bad, considering no optimization
- Goal is to reach interactive speed

* HOSVD: Higher Order Singular Value Decomposition

Higher Order SVD for a 3-D Array

$$\mathcal{A} = \mathcal{S} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$$

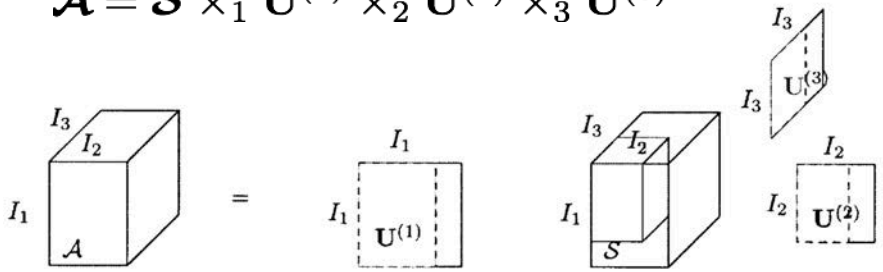
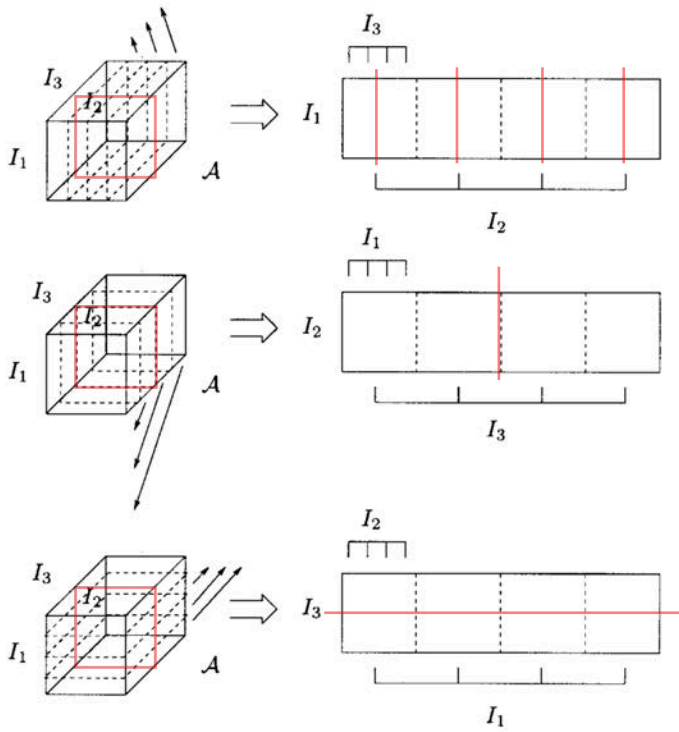


FIG. 4. *Visualization of the HOSVD for a third-order tensor.*

Source: Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl*, 21:1253-1278, 2000. (including figures)

$(I_1 \times I_2 \times I_3) = (41 \times 32 \times 232{,}011)$



$$\mathbf{A}_{(1)} = \mathbf{U}^{(1)}\mathbf{\Sigma}^{(1)}\mathbf{V}^{(1)T}$$

"wide & skinny column-block" = `tshaq`

$$\mathbf{A}_{(2)} = \mathbf{U}^{(2)}\mathbf{\Sigma}^{(2)}\mathbf{V}^{(2)T}$$

"wide & skinny column-block" = `tshaq`

$$\mathbf{A}_{(3)} = \mathbf{U}^{(3)}\mathbf{\Sigma}^{(3)}\mathbf{V}^{(3)T}$$

"tall & skinny row-block" = `shaq`

Keep $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(3)}$ and compute $\mathcal{S} = \mathcal{A} \times_1 \mathbf{U}^{(1)T} \times_2 \mathbf{U}^{(2)T} \times_3 \mathbf{U}^{(3)T}$

Note that $(\mathcal{A} \times_i \mathbf{U}^{(i)})_{(i)} = \mathbf{U}^{(i)T}\mathbf{A}_{(i)}$

So that $\mathbf{S}_{(3)} = \mathbf{U}^{(3)T}(\mathbf{U}^{(2)T}(\mathbf{U}^{(1)T}\mathbf{A}_{(1)})_{(2)})_{(3)}$

Familiar Code but Cognizant of Distributed Dimension

$$\mathbf{A}_{(1)} = \mathbf{U}^{(1)}\mathbf{\Sigma}^{(1)}\mathbf{V}^{(1)T}$$
$$\mathbf{A}_{(2)} = \mathbf{U}^{(2)}\mathbf{\Sigma}^{(2)}\mathbf{V}^{(2)T}$$
$$\mathbf{A}_{(3)} = \mathbf{U}^{(3)}\mathbf{\Sigma}^{(3)}\mathbf{V}^{(3)T}$$

Keep $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(3)}$ and compute the $\boldsymbol{\mathcal{S}}$

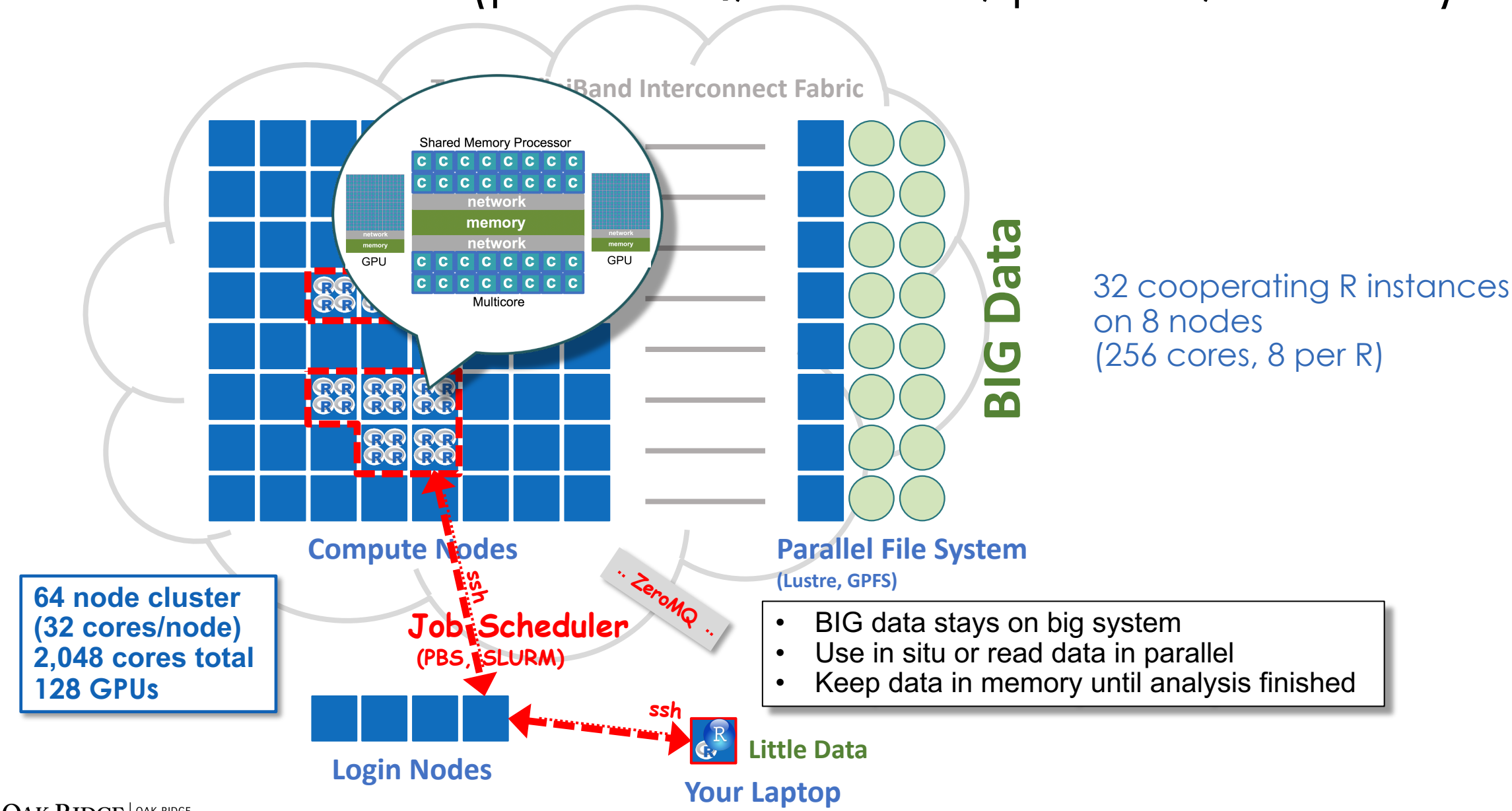
$$\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{A}} \times_1 \mathbf{U}^{(1)T} \times_2 \mathbf{U}^{(2)T} \times_3 \mathbf{U}^{(3)T}$$

Note that $(\boldsymbol{\mathcal{A}} \times_i \mathbf{U}^{(i)})_{(i)} = \mathbf{U}^{(i)T}\mathbf{A}_{(i)}$

So $\mathbf{S}_{(3)} = \mathbf{U}^{(3)T}(\mathbf{U}^{(2)T}(\mathbf{U}^{(1)T}\mathbf{A}_{(1)})_{(2)})_{(3)}$

```
library(kazaam)
tens = read_xgc_window(file_var, var, w_center, window)$Data
tdim = dim(tens) # local dimensions (1, 2, 3d)

## A1 unfolding and SVD
A1 = as.vector(tens) # (1*2*3d)
dim(A1) = c(tdim[1], tdim[2]*tdim[3]) # (1, 2*3d)
U1 = svd(tshaq(A1))$u # (1, 2*3d) tshaq => (1, 1)

re_ord = . . . # index for 1*2 => 2*1
re_ord_inv = . . . # index for 2*1 => 1*2
## A2 unfolding and SVD
. . .
U2 = . . .
## A3 unfolding and SVD
. . .
U3 = . . .

## Core tensor computation
S1 = crossprod(U1, A1) # t((1, 1)) %*% (1, 2*3d)
S1 = as.vector(S1) # (1*2*3d)
dim(S1) = c(tdim[1]*tdim[2], tdim[3]) # (1*2, 3d)

S2 = S1[re_ord, ] # (1*2, 3d) => (2*1, 3d)
dim(S2) = c(tdim[2], tdim[1]*tdim[3]) # (2, 1*3d)
S2 = crossprod(U2, S2) # t((2, 2)) %*% (2, 1*3d)

S3 = as.vector(S2) # (2, 1, 3d)
dim(S3) = c(tdim[2]*tdim[1], tdim[3]) # (2*1, 3d)
S3 = S3[re_ord_inv, ] # (2*1, 3d) => (1*2, 3d)
S3 = crossprod(U3, shaq(t(S3))) # t((3d, 3c)) %*% t((1*2, 3d)) shaq = (3c, 1*2)

S = t(S3) # dim (3c, 1*2) => (1*2, 3c)
dim(S) = c(tdim[1], tdim[2], min(tdim[1]*tdim[2], allreduce(tdim[3])))
```

Interactive SPMD (pbdZMQ, remoter, pbdCS, launchr)



32 cooperating R instances
on 8 nodes
(256 cores, 8 per R)

- BIG data stays on big system
- Use in situ or read data in parallel
- Keep data in memory until analysis finished

pbdR Project Support

Funding:

ADW Group, Oak Ridge Leadership Computing Facility, a ***Department of Energy Office of Science*** *User Facility supported under Contract DE-AC05-00OR22725*

U.S. Veterans Administration*, Million Veterans Program, Scientific Computing Research Support*

National Science Foundation Division of Mathematical Sciences *Grant No. 1418195, 2014-2019.*

National Science Foundation *National Institute for Mathematical and Biological Synthesis, under Award No. EF-0832858 and DBI-1300426, 2013-2014.*

National Science Foundation *Division of Molecular and Cellular Biosciences Award MCB-1120370, 2013-2014.*

National Science Foundation *Office of Cyberinfrastructure under Award No. ARRA-NSF-OCI-0906324: Remote Data Analysis and Visualization center, 2012-2013.*

Office of Biological and Environmental Research, U.S. ***Department of Energy Office of Science*** *under Contract No. DE-AC05-00OR22725: Visual Data Exploration and Analysis of Ultra-large Climate Data, 2011-2013.*

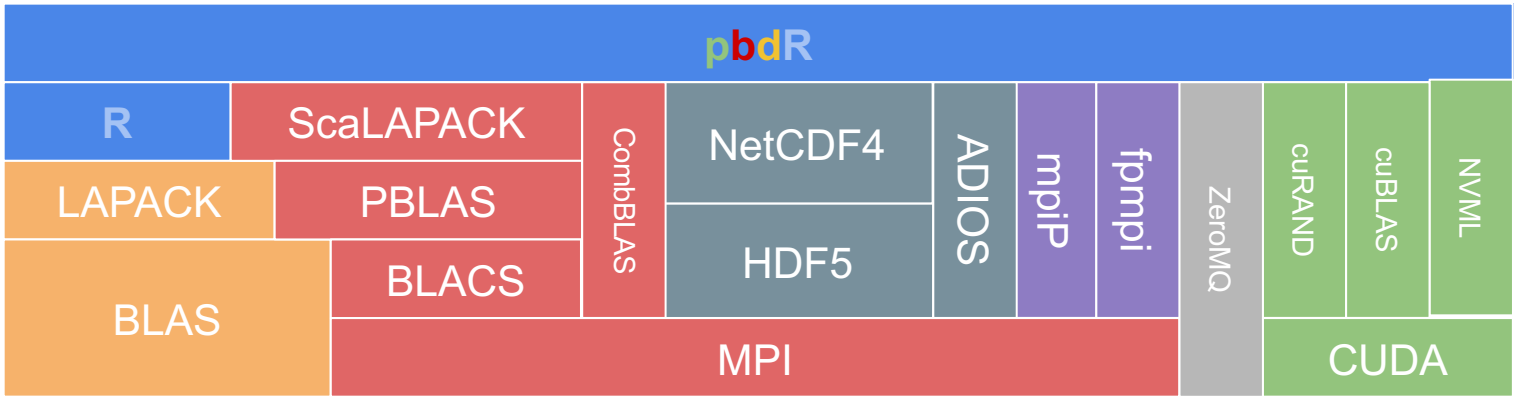
Resources:

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the ***Office of Science of the U.S. Department of Energy*** *under Contract No. DE-AC05-00OR22725.*

This research used resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, which is supported by the **Office of Science of the U.S. Department of Energy** under Contract No. DE-AC05-00OR22725.

This research used resources of the National Institute for Computational Sciences at the University of Tennessee, Knoxville, which is supported by the ***Office of Cyberinfrastructure of the U.S. National Science Foundation****.*

Thank You!



pbdR
Programming with Big Data in R

Distinctions

- A component of 2019 Cray® Urika®-CS AI and Analytics Suite
- Intel ® HPC Developer Conference 2017: People's Choice Award, High Productivity Languages Track
- 2016 ORNL Significant Event Award
- Benchmark code for Frontier procurement

ORNL Installations

- Supported software on OLCF platforms (Rhea, Eos, Titan, Summit)
- Available on CADES resources
- Install from: https://pbdR.org/releases

Schmidt, Chen, Matheson, and Ostrouchov (2017). Programming with BIG Data in R: Scaling Analytics from One to Thousands of Nodes, ***Big Data Research***, *vol.8, p.1-11*.

Schmidt, Chen, and Ostrouchov (2016). Introducing a New Client/Server Framework for Big Data Analytics with the R Language. ***XSEDE16 Conference on Diversity, Big Data, and Science at Scale***, Article No. 38.