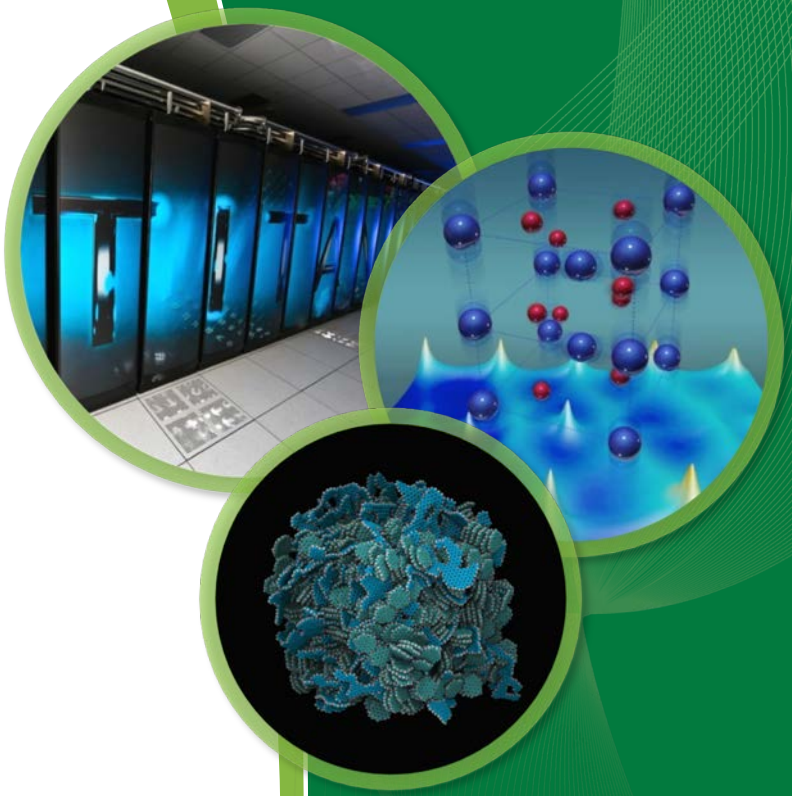
Stalking the Interactive Terabyte with R

George Ostrouchov[1,2,3]
Drew Schmidt[2]
Michael Matheson[2]

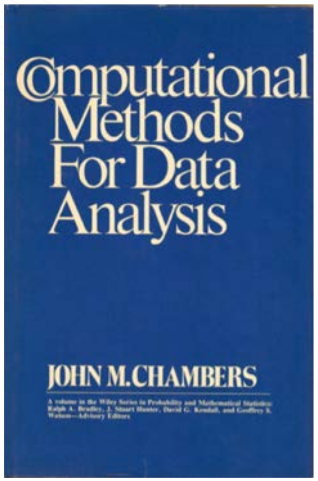
[1]Scientific Data Group, CSMD
[2]Advanced Data and Workflow Group, NCCS
Oak Ridge National Laboratory

[3]Business Analytics and Statistics, UTK (Joint Faculty)
University of Tennessee, Knoxville

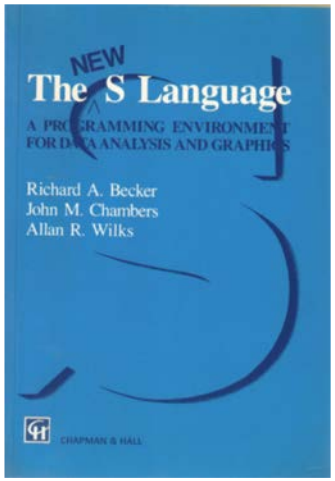


Intel® HPC Developer Conference 2017

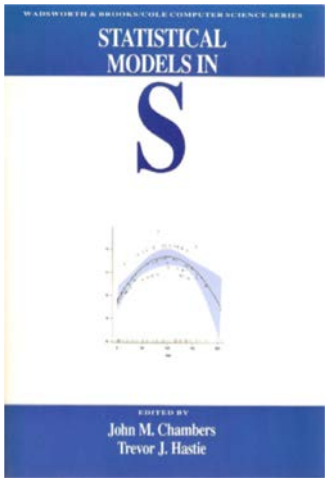
From Methods and Batch Runs to Interactive Software for Data Analysis



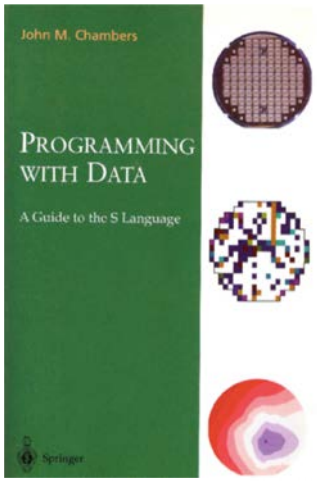
Chambers.
Computational Methods for Data Analysis.
Wiley, 1977.



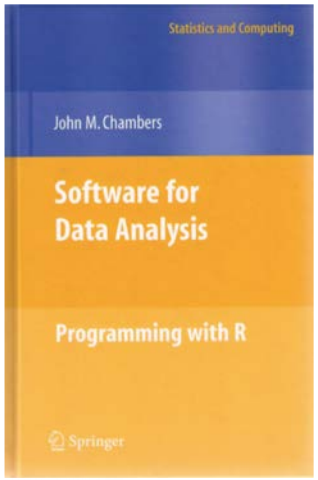
Becker, Chambers, and Wilks.
The New S Language.
Chapman & Hall, 1988.



Chambers and Hastie.
Statistical Models in S.
Chapman & Hall, 1992.



Chambers.
Programming with Data.
Springer, 1998.

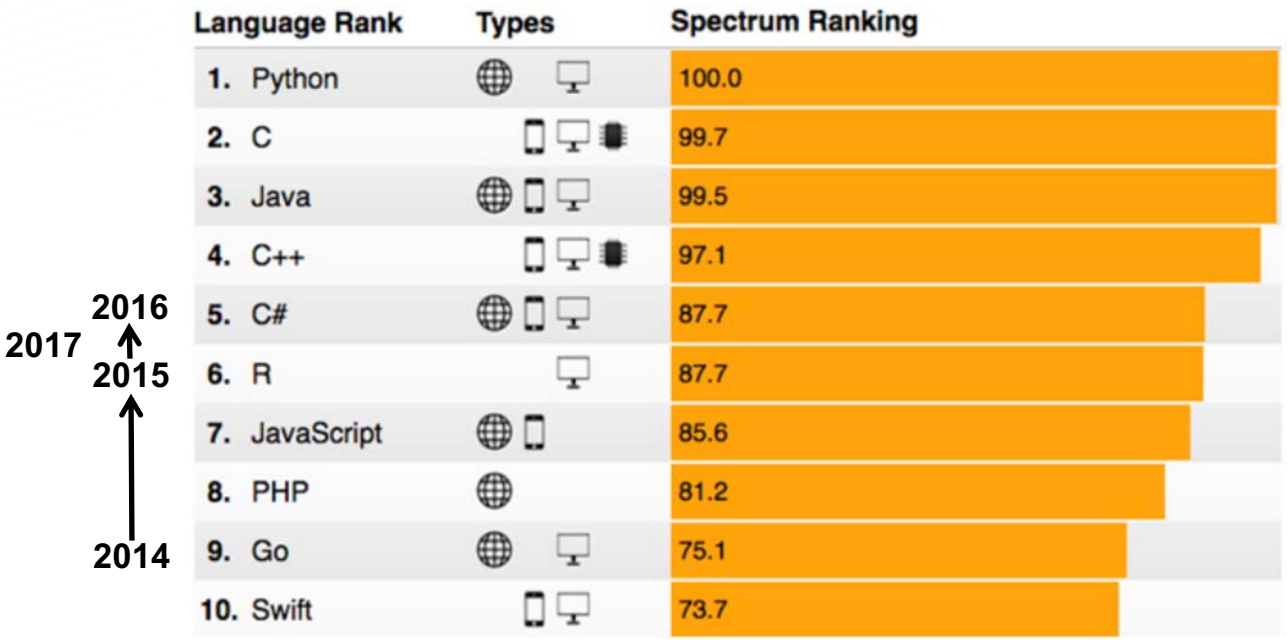


Chambers.
Software for Data Analysis: Programming with R.
Springer, 2008.

Thanks to Dirk Eddelbuettel for this slide idea and to John Chambers for providing the high-resolution scans of the covers of his books.

The R Software for Data Approaches Popularity of General Programming Languages

2017 IEEE Spectrum's Ranking of Programming Languages



Discovery Thrives on Diversity

Resources for Learning R

- RStudio IDE
- Favorite book: The Art of R Programming by Norm Matloff:
- Short intro: R Language for Programmers by John Cook
- Longer intro: aRrgh: a newcomer's (angry) guide to R, by Tim Smith and Kevin Ushey
- Grammar of graphics ggplot2, tidyverse for data wrangling, and Advanced R by Hadley Wickham
- Help: Mailing list archives and the [R] stackoverflow tag.
- Distributed programming with big data in R: pbdR.org, and our paper in *Big Data Research* (Schmidt et al., 2016).

pbdR Project

Current Developers

Wei-Chen Chen, USFDA

Michael Matheson, ORNL

George Ostrouchov, ORNL & UTK

Drew Schmidt, ORNL

Past Developers and Contributors

Christian Heckendorf, Yuping Lu, Pragneshkumar Patel, Gaurav Sehrawat, Whit Armstrong, Ewan Higgs, Michael Lawrence, David Pierce, Brian Ripley, ZhaoKang Wang, Hao Yu

Support

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. ("supported software" at OLCF)

National Science Foundation Division of Mathematical Sciences under Grant No. 1418195, 2014-2018.

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

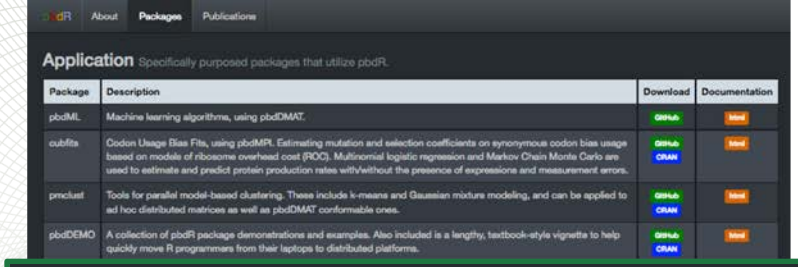
The Division of Molecular and Cellular Biosciences, National Science Foundation Award MCB-1120370, 2013-2014.

The Office of Cyberinfrastructure of the U.S. National Science Foundation under Award No. ARRA-NSF-OCI-0906324 for NICS-RDAV center, 2012-2013.

U.S. Department of Energy Office of Science under Contract No. DE-AC05-00OR22725, 2011-2013.

Also 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.

pbdr.org/packages

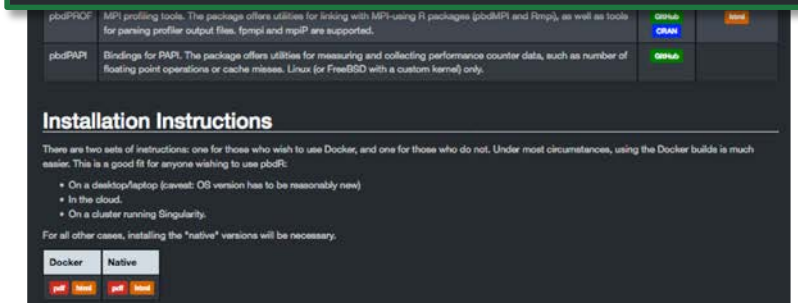


Computation Packages frameworks for building other scalable tools.

Package	Description	Download	Documentation
pbdDMAT	Distributed matrix classes and methods. The package includes numerous methods for manipulating and reshaping distributed matrices, as well as linear algebra and statistics routines. Through extensive use of R's S4 methods, these functions have identical syntax to serial R.	GitHub CRAN	html
pbdMPI	A high-level interface to MPI. The package handles linking issues, as well as offers a very simple R interface for MPI programming.	GitHub CRAN	html

Developers Tools for developers.

Package	Description	Download	Documentation
pbdTEST	A testing framework for pbdR.	GitHub	
pbdBASE	Base utilities for distributed matrices. For advanced programmers only.	GitHub CRAN	html
pbdSLAP	The Scalable Linear Algebra Package. A distribution of ScaLAPACK, this package greatly simplifies package build and linking issues for distribute matrix programming.	GitHub CRAN	html



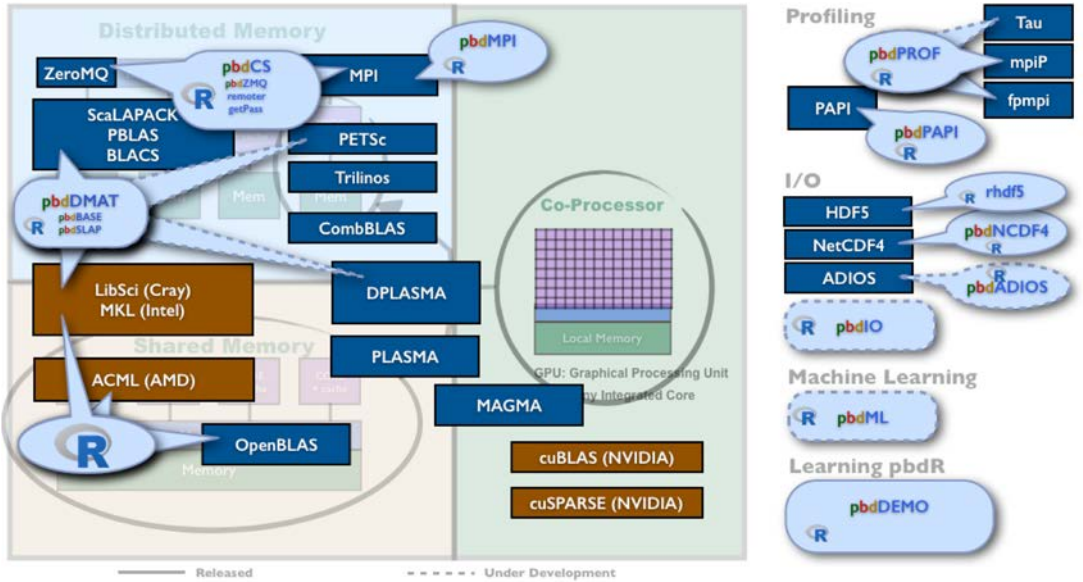
OAK RIDGE National Laboratory

A Scalable Platform for Developing Interactive Big Data Analytics

pbdR
Programming with Big Data in R

http://pbdr.org

- Engage parallel math libraries at scale
- R language unchanged
- New distributed concepts
- New profiling capabilities
- New interactive SPMD parallel
- In-situ distributed capability
- In-situ staging capability via ADIOS
- 2016 ORNL Significant Event Award



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: "*several hours on . . . Apache Spark, . . . less than a minute using R.*"

Modern statistical algorithm + pbdR infrastructure + HPC Libraries + HPC Hardware

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

Implementing Modern Statistical Algorithms: Truncated SVD from Random Projections

Prototype for Randomized SVD

Given an $m \times n$ matrix $\boldsymbol{A}$, a target number k of singular vectors, and an exponent q (say, $q = 1$ or $q = 2$), this procedure computes an approximate rank-$2k$ factorization $\boldsymbol{U\Sigma V}^*$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal, and $\boldsymbol{\Sigma}$ is nonnegative and diagonal.

Stage A:

1. Generate an $n \times 2k$ Gaussian test matrix $\boldsymbol{\Omega}$.
2. Form $\boldsymbol{Y} = (\boldsymbol{AA}^*)^q \boldsymbol{A\Omega}$ by multiplying alternately with $\boldsymbol{A}$ and $\boldsymbol{A}^*$.
3. Construct a matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$.

Stage B:

4. Form $\boldsymbol{B} = \boldsymbol{Q}^* \boldsymbol{A}$.
5. Compute an SVD of the small matrix: $\boldsymbol{B} = \widetilde{\boldsymbol{U}} \boldsymbol{\Sigma} \boldsymbol{V}^*$.
6. Set $\boldsymbol{U} = \boldsymbol{Q}\widetilde{\boldsymbol{U}}$.

Note: The computation of $\boldsymbol{Y}$ in step 2 is vulnerable to round-off errors. When high accuracy is required, we must incorporate an orthonormalization step between each application of $\boldsymbol{A}$ and $\boldsymbol{A}^*$; see Algorithm 4.4.

Algorithm 4.4: Randomized Subspace Iteration

Given an $m \times n$ matrix $\boldsymbol{A}$ and integers ℓ and q, this algorithm computes an $m \times \ell$ orthonormal matrix $\boldsymbol{Q}$ whose range approximates the range of $\boldsymbol{A}$.

1. Draw an $n \times \ell$ standard Gaussian matrix $\boldsymbol{\Omega}$.
2. Form $\boldsymbol{Y}_0 = \boldsymbol{A\Omega}$ and compute its QR factorization $\boldsymbol{Y}_0 = \boldsymbol{Q}_0 \boldsymbol{R}_0$.
3. **for** $j = 1, 2, \ldots, q$
4. Form $\widetilde{\boldsymbol{Y}}_j = \boldsymbol{A}^* \boldsymbol{Q}_{j-1}$ and compute its QR factorization $\widetilde{\boldsymbol{Y}}_j = \widetilde{\boldsymbol{Q}}_j \widetilde{\boldsymbol{R}}_j$.
5. Form $\boldsymbol{Y}_j = \boldsymbol{A}\widetilde{\boldsymbol{Q}}_j$ and compute its QR factorization $\boldsymbol{Y}_j = \boldsymbol{Q}_j \boldsymbol{R}_j$.
6. **end**
7. $\boldsymbol{Q} = \boldsymbol{Q}_q$.

Serial R

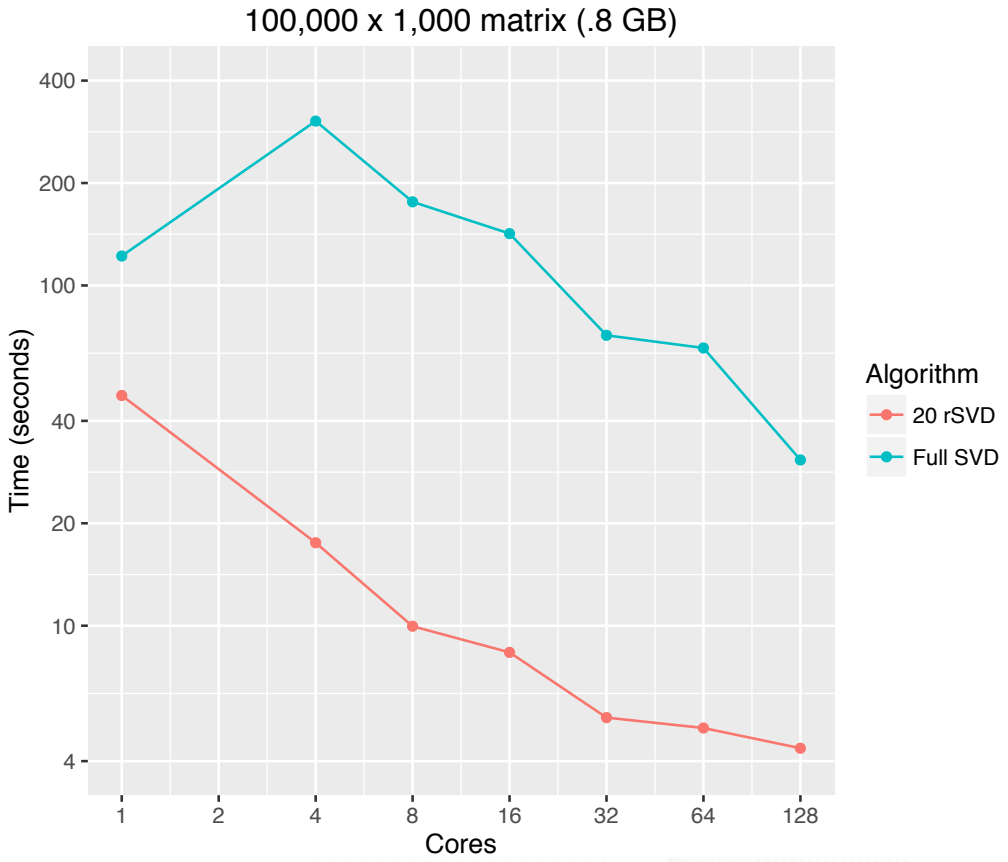
```
1  randSVD <- function(A, k, q=3)
2    {
3      ## Stage A
4      Omega <- matrix(rnorm(n*2*k),
                 nrow=n, ncol=2*k)
5      Y <- A %*% Omega
6      Q <- qr.Q(qr(Y))
7      At <- t(A)
8      for(i in 1:q)
9        {
10         Y <- At %*% Q
11         Q <- qr.Q(qr(Y))
12         Y <- A %*% Q
13         Q <- qr.Q(qr(Y))
14       }
15
16     ## Stage B
17     B <- t(Q) %*% A
18     U <- La.svd(B)$u
19     U <- Q %*% U
20     U[, 1:k]
21   }
```

[1] Halko, Martinsson, and Tropp. 2011. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions *SIAM Review* **53** 217–288

From Serial Pseudocode to Scalable Code and Benchmark Data in One Day!

Parallel pbdR

```
1  randSVD <- function(A, k, q=3)
2    {
3      ## Stage A
4      Omega <- ddmatrix("rnorm", nrow=n, ncol=2*k)
5      Y <- A %*% Omega
6      Q <- qr.Q(qr(Y))
7      At <- t(A)
8      for(i in 1:q)
9        {
10         Y <- At %*% Q
11         Q <- qr.Q(qr(Y))
12         Y <- A %*% Q
13         Q <- qr.Q(qr(Y))
14       }
15
16     ## Stage B
17     B <- t(Q) %*% A
18     U <- La.svd(B)$u
19     U <- Q %*% U
20     U[, 1:k]
21   }
```



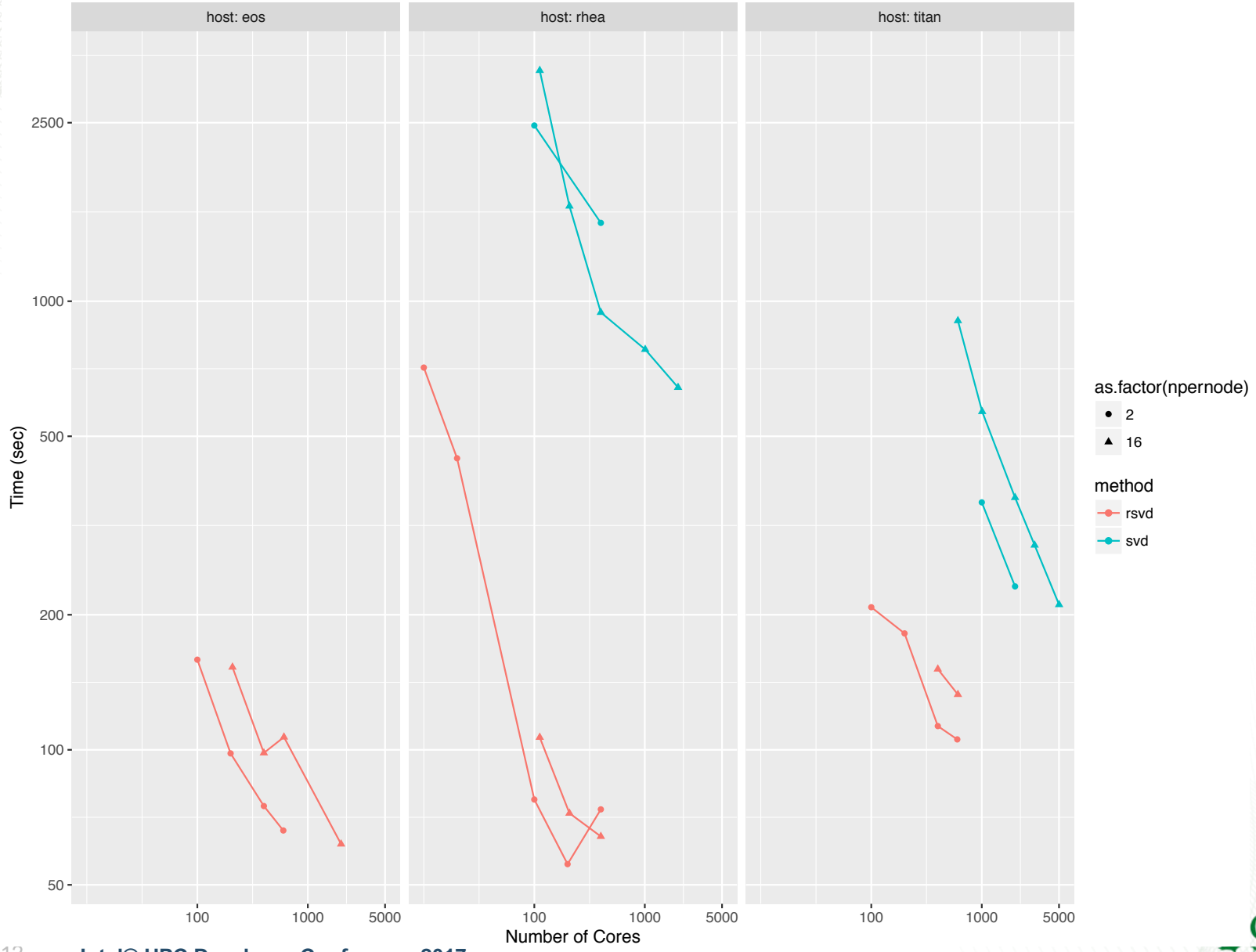
OAK RIDGE National Laboratory

Microscopy Data PCA Benchmark (134 GB) with rsvd

fastpca.R

```
1  suppressPackageStartupMessages(library(rhdf5))
2  suppressPackageStartupMessages(library(pbdIO))
3  suppressPackageStartupMessages(library(pbdML))
4  init.grid( )
5
6  var  <- "your_hdf5_full_variable_name"
7  h5f <- H5Fopen( "your_hdf5_file_name" )
8  h5d <- H5Dopen( h5f, var )
9  h5s <- H5Dget_space( h5d )
10 dims <- H5Sget_simple_extent_dims( h5s )$size
11 rows <- dims[2]   # row-major to column-major
12 cols <- dims[1]   # because data written by C/Py
13
14 ## read C/Py-written blocks of rows into R blocks of columns
15 my_rows <- comm.chunk(rows, form="vector", type="equal")
16 A <- t(h5read(h5f, var, index=list(NULL, my_rows)))
17
18 ## add glue to make a global column-block ddmatrix
19 A <- new("ddmatrix", Data=A, dim=c(rows, cols), ldim=dim(A), bldim=dim(A), ICTXT=2)
20
21 ## rearrange into block-cyclic
22 A <- as.blockcyclic( A )
23
24 ## get 32 top singular values and vectors
25 Res <- rpca( A, k = 32 )
26
27 ## print the singular values
28 comm.print( Res$d )
29
30 finalize()
```

Microscopy Data PCA Benchmark (134 GB)



New Developments for Skinny Matrices: `shaq` Matrix in `kazaam` Package

- For skinny matrices, horizontal or vertical
- Especially useful when only top singular vectors needed
- Relies on allreduce and a small SVD
- Interactive speeds for TB-size data

$m \times n \qquad X = UDV^T$

$m \gg n \qquad X^T X = VDU^T UDV^T = VD^2V^T$ *n × n*

$U = XVD^{-1}$

$m \ll n \qquad XX^T = UDV^T VDU^T = UD^2U^T$ *m × m*

$V^T = D^{-1}U^T X$

Tall – Distributed by Row Blocks

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_r \end{bmatrix}$$

$$X^T X = \sum_{i=1}^{r} X_i^T X_i = VD^2V^T$$

Algorithm 1 Distributed Crossproduct

1: Given distributed matrix X with local sub-data `X.local`, produce $X^T X$
2: **function** CROSSPROD(X)
3: `cp_local <- transpose(X.local) * X.local`
4: `cp_global <- MPI_allreduce(cp_local)`
5: `return cp_global`
6: **end function**

Algorithm 2 Distributed SVD

1: Given distributed matrix X, factor $X = U\Sigma V^T$
2: **function** SVD(X)
3: `cp_global <- crossprod(X)`
4: Σ `<- sqrt(eigenvalues(cp_global))`
5: V `<- eigenvectors(cp_global)`
6: U `<- X.local * V *` Σ^{-1}
7: return Σ, U, V
8: **end function**

$$U = XVD^{-1} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_r \end{bmatrix} VD^{-1} = \begin{bmatrix} X_1VD^{-1} \\ X_2VD^{-1} \\ \vdots \\ X_rVD^{-1} \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_r \end{bmatrix}$$

Wide – Distributed by Column Blocks

$$X = [X_1|X_2|\cdots|X_r]$$

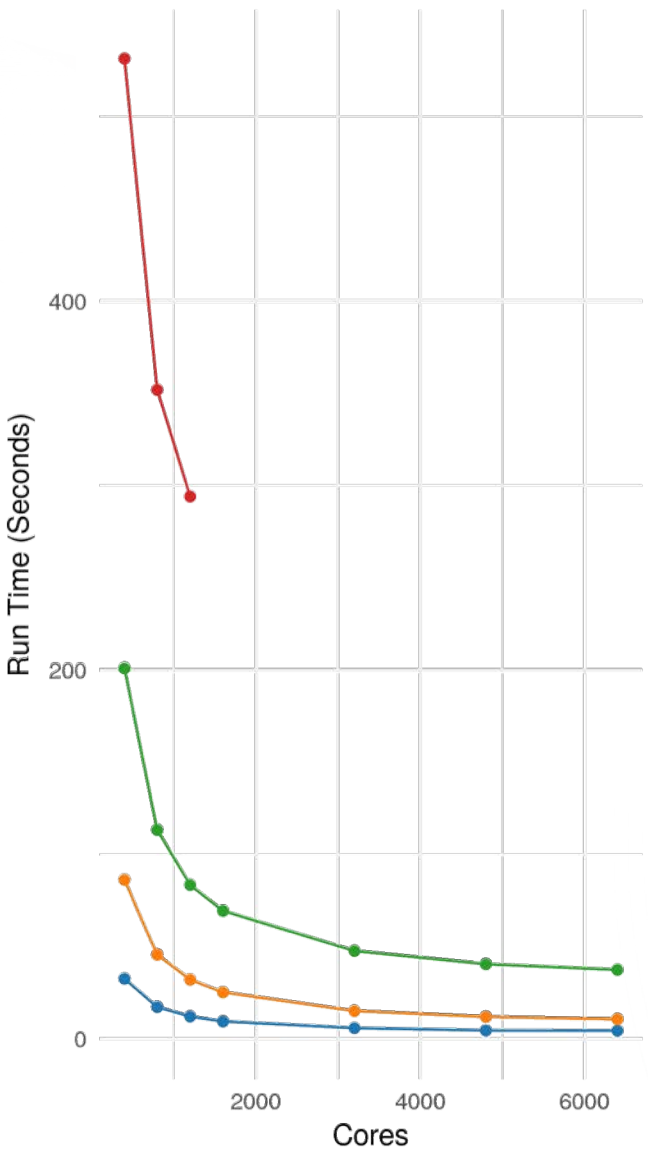
$$XX^T = [X_1|X_2|\cdots|X_r]\begin{bmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_r^T \end{bmatrix} = \sum_{i=1}^{r} X_i X_i^T = UD^2U^T$$

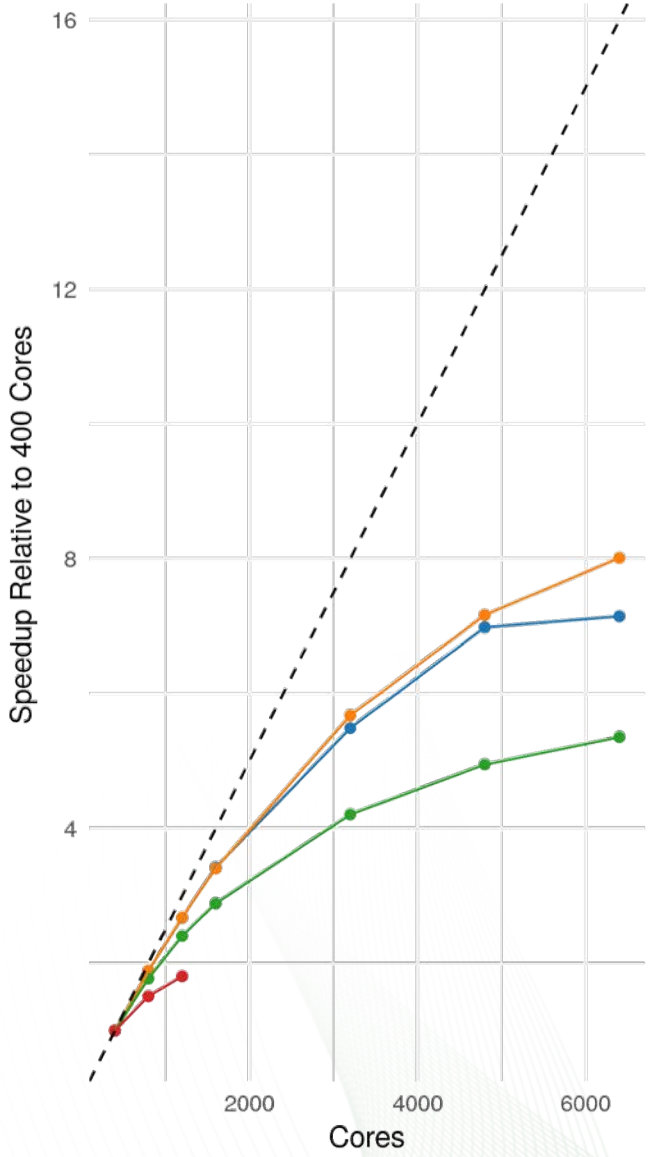
$$\begin{aligned} V^T &= D^{-1}U^T X \\ &= D^{-1}U^T [X_1|X_2|\cdots|X_r] \\ &= \left[D^{-1}U^T X_1|D^{-1}U^T X_2|\cdots|D^{-1}U^T X_r\right] \\ &= \left[V_1^T|V_2^T|\cdots|V_r^T\right] \end{aligned}$$

Interactive Speeds with pbdR on KNL Cluster

Tall and Skinny
300 GiB Matrix SVD

Cray XC40 with KNL

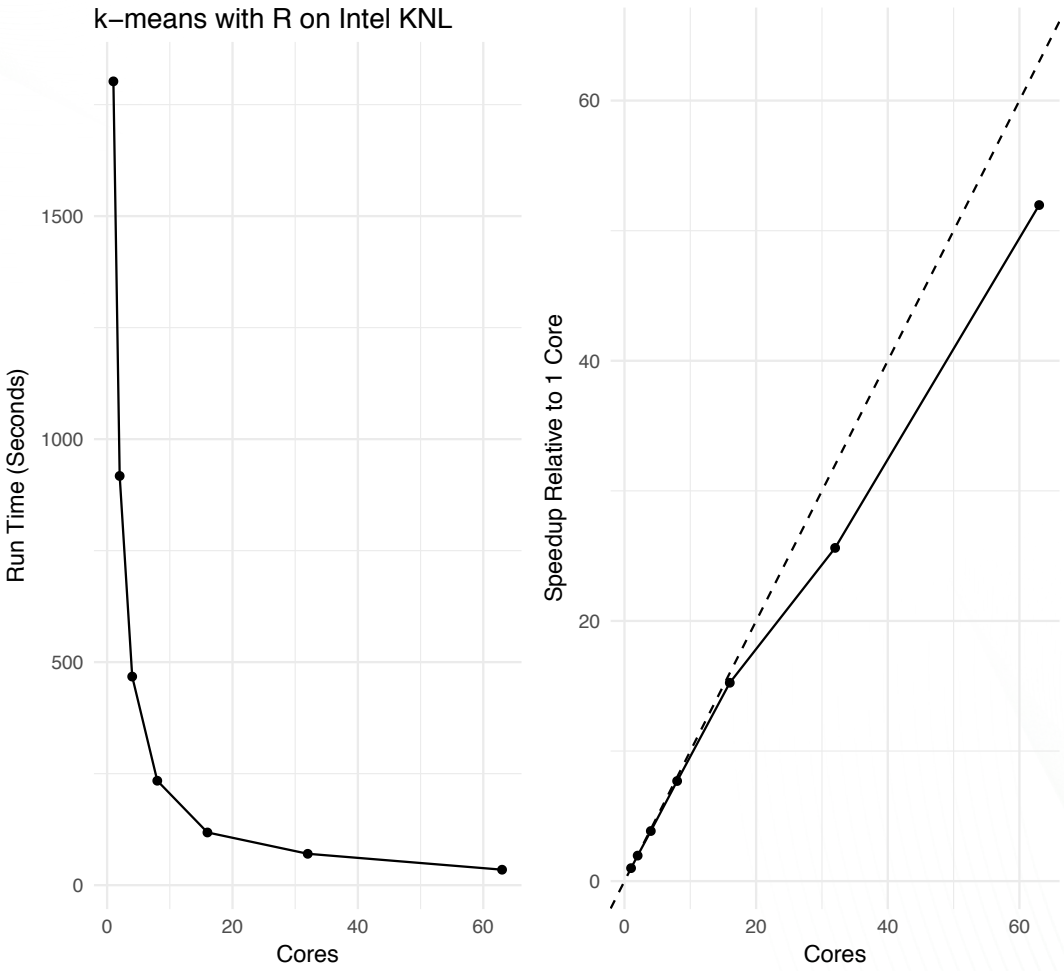




Number of Columns 625 1250 2500 5000

OAK RIDGE National Laboratory

K-means Experiments on One KNL Node



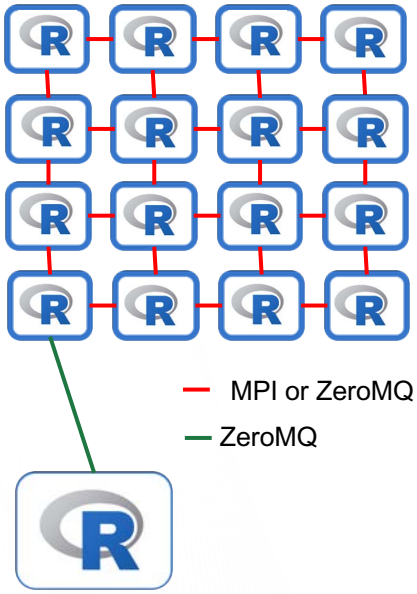
Performance Analysis Tools for R

- pbdPROF
  - Brings fpmpi and mpiP profiling to R code
- pbdPAPI
  - Brings PAPI (and IPCM) capabilities to measure R code
- hpcvis
  - Graphics and analytics for fpmpi and mpiP data
  - Graphics and analytics for pbdPAPI data objects

- Schmidt, Chen, Heckendorf, and Ostrouchov (2017) Analyzing Analytics: Advanced Performance Analysis Tools for R

New “Interactive SPMD” Client-Server

- pbdZMQ
  - Wraps and provides ease of use for ZeroMQ communication library
  - Includes a sufficient ZeroMQ distribution
- remoter
  - control a remote R session from a local one
  - Based on pbdZMQ
  - Can use a relay
- getPass
  - A portable way to read user input with masking
- pbdCS
  - Utilities for interactive SPMD/MPI programming from an R session



Thank you!

pbdR.org

OAK RIDGE National Laboratory