pbdR: Bringing R Analytics to Large Distributed Architectures

George Ostrouchov
Oak Ridge National Laboratory, Oak Ridge, TN
and
University of Tennessee, Knoxville, TN

Swiss Supercomputing Center
Lugano, June 20, 2014

The pbdR Core Team

Wei-Chen Chen[1]
George Ostrouchov[2,3]
Pragneshkumar Patel[3]
Drew Schmidt[3]

Support

This work used resources of 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 under Award No. ARRA-NSF-OCI-0906324 for NICS-RDAV center. This work also 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.

[1]Department of Ecology and Evolutionary Biology
University of Tennessee, Knoxville TN, USA

[2]Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge TN, USA

[3] Joint Institute for Computational Sciences
University of Tennessee, Knoxville TN, USA

Contents

What is R?

- *Lingua franca* for data analytics and statistical computing
- Part programming language, part data analysis package, syntax designed for data
- Dialect of S (Bell Labs, 1976)
- Licensed under GPL in 1995
- 2+ million users, dominates new work in Statistics
- Switzerland: highest per-capita use of R!

`http://blog.revolutionanalytics.com/2014/04/a-world-map-of-r-user-activity.html`

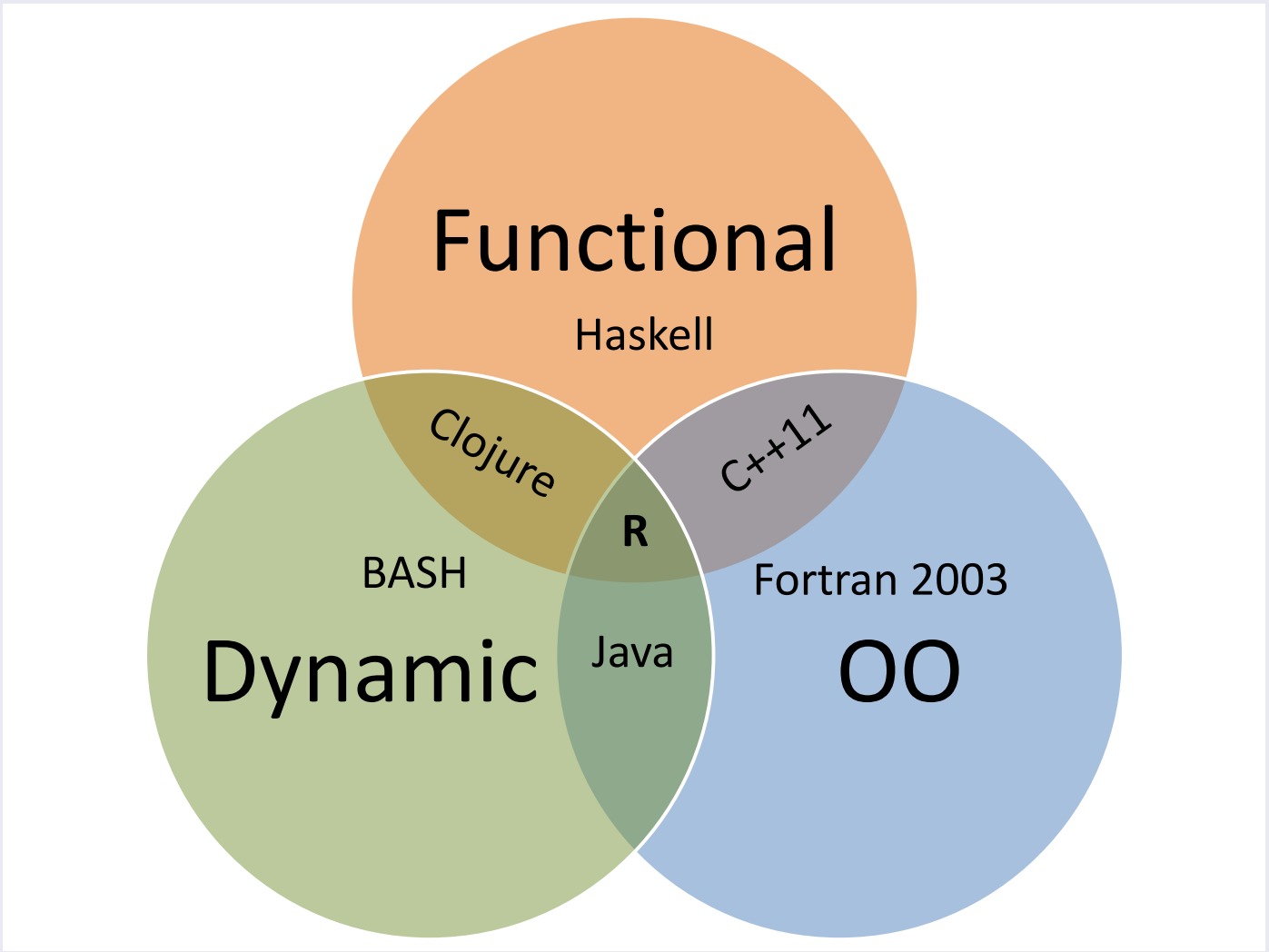
Who uses R?


ORACLE
MERCK
KICKSTARTER
Bank of America
okcupid
The New York Times
Shell
Pfizer
ORBITZ
mozilla FOUNDATION
LLOYD'S
Google
ebay
bing
Ford

Who integrates R?

- SAS
- Tableau
- IBM-SPSS
- VTK (Paraview, VisIt)
- alteryx
- Omniscope
- Python
- Java
- ORACLE
- KNIME
- XCEL
- ...

Language Paradigms



Dynamic Data Types

- Storage: logical, int, double, double complex, character
- Structures: vector, matrix, array, list, dataframe
- Caveats: (Logical) `TRUE`, `FALSE`, `NA`

High Level Syntax

```
1 x <- matrix(rnorm(30), nrow=10)
2 x <- x[-1, 2:5]
3 xtx <- t(x) %*% x
4 xtx <- crossprod(x)
5 ans <- prcomp(log(abs(x) + 1))
6 ans <- glm(y ~ a + b + ns(u,12), binomial(), mydata)
7 plot(ans)
8 print(ans)
```

CRAN Task View (5,000+ packages) http://cran.rstudio.com/web/views/

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
NumericalMathematics	Numerical Mathematics
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
SpatioTemporal	Handling and Analyzing Spatio-Temporal Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis
WebTechnologies	Web Technologies and Services
gR	gRaphical Models in R

Other R pakage repositories

- 

  - Tools for the analysis and comprehension of high-throughput genomic data
  - 800+ packages `http://www.bioconductor.org`
- Rmetrics
  - 30+ packages for financial computing `https://rmetrics.org` (ETH)
- ...

Why take R to HPC?

1. People love it
2. Highly extensible
3. Unmatched diversity of analytical methods
4. Thorough treatment of uncertainty throughout
5. Gold standard for graphics (traditional, grid, and ggplot2)
6. HPC community needs a high-level language for data analysis

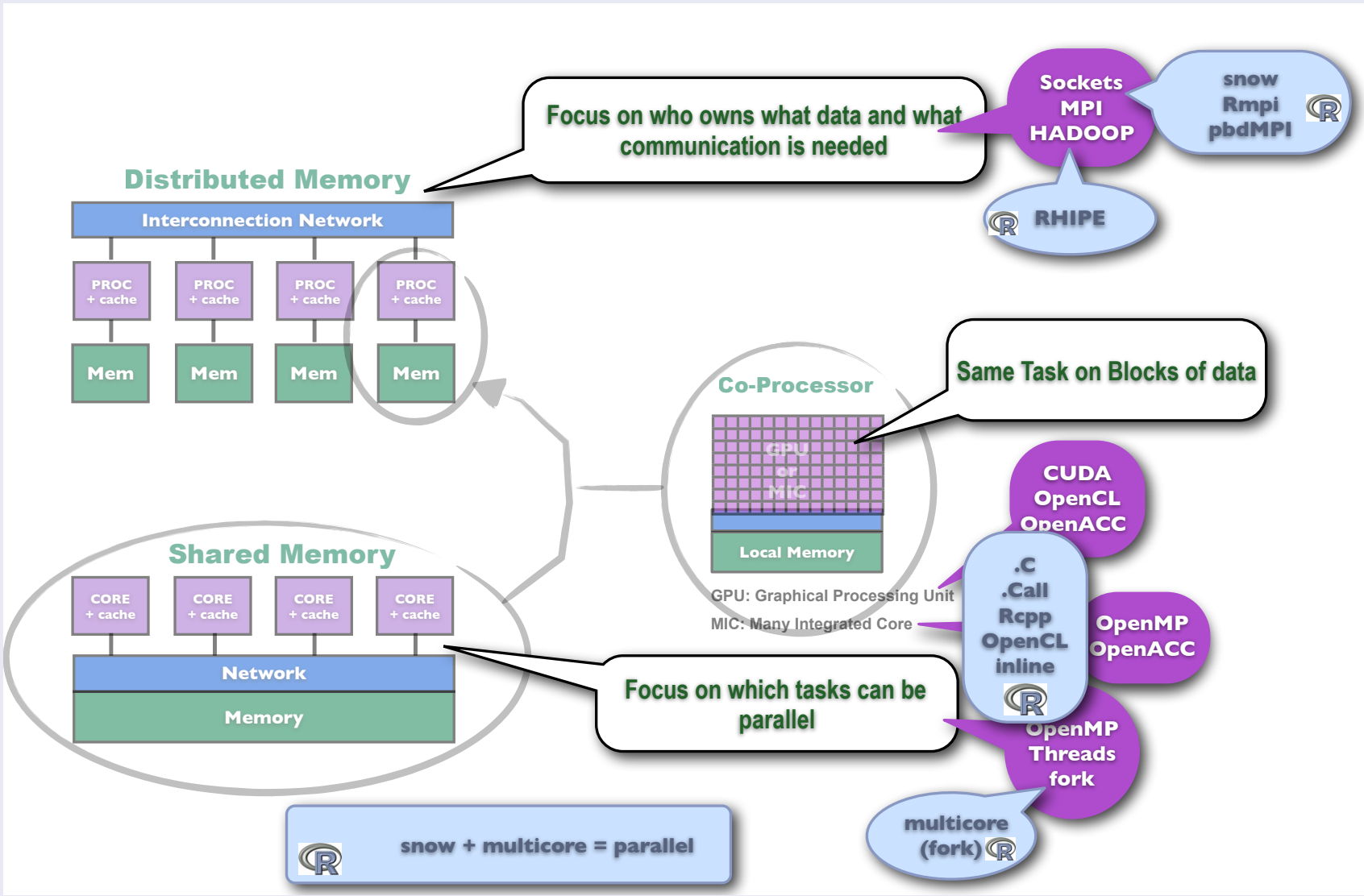
Data analysis is interactive!

- Data reduction to knowledge
- S (and R) interactive "answer" to batch data analysis
- Efficient use of expensive people
- Iterative process with same data
  - Diagnostics of fit
  - Quantification of uncertainty
  - Interpretation of results

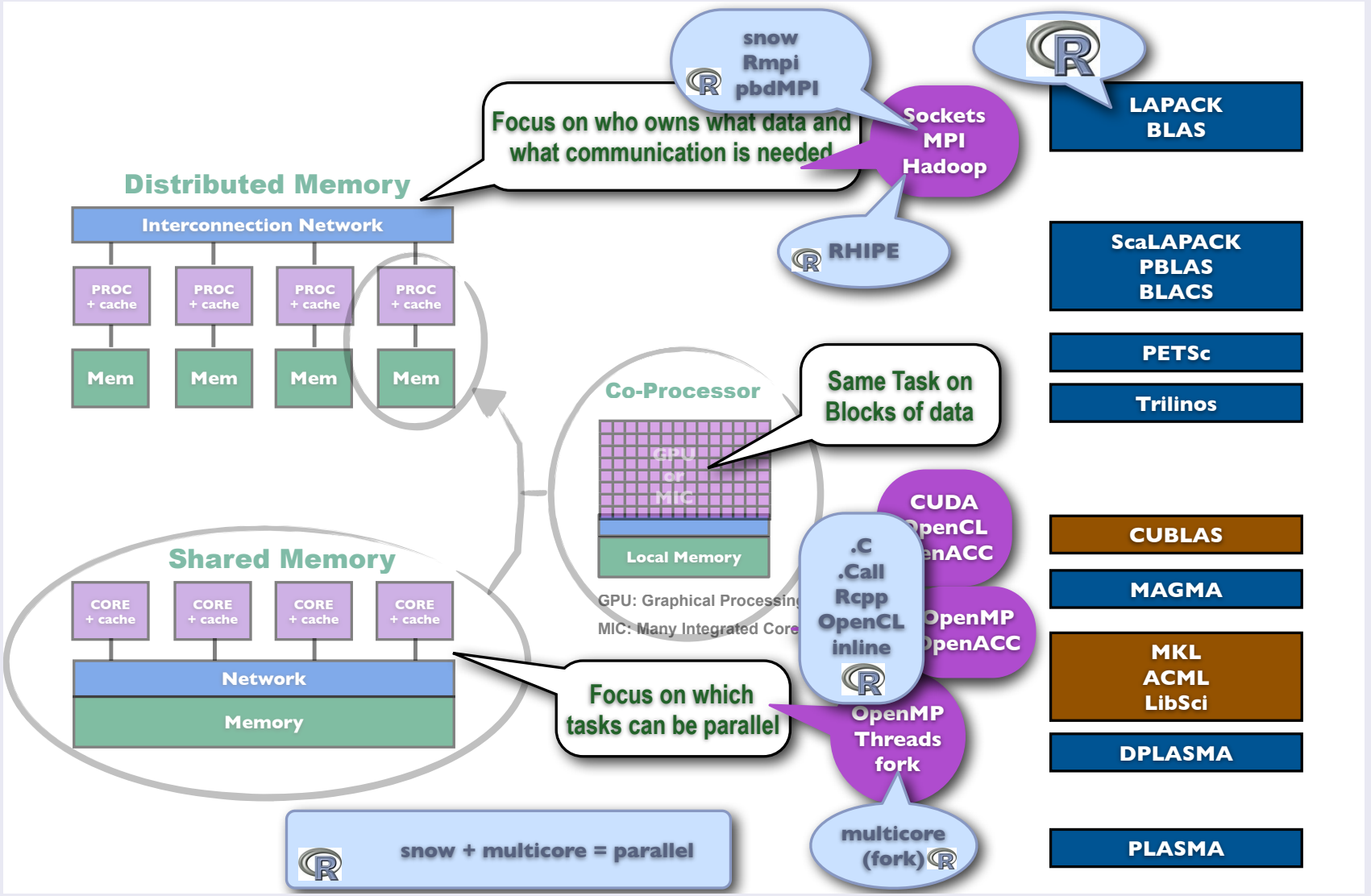
HPC is batch!

- Traditionally data generation
- Efficient use of expensive platforms
- Long time to assimilate generated data

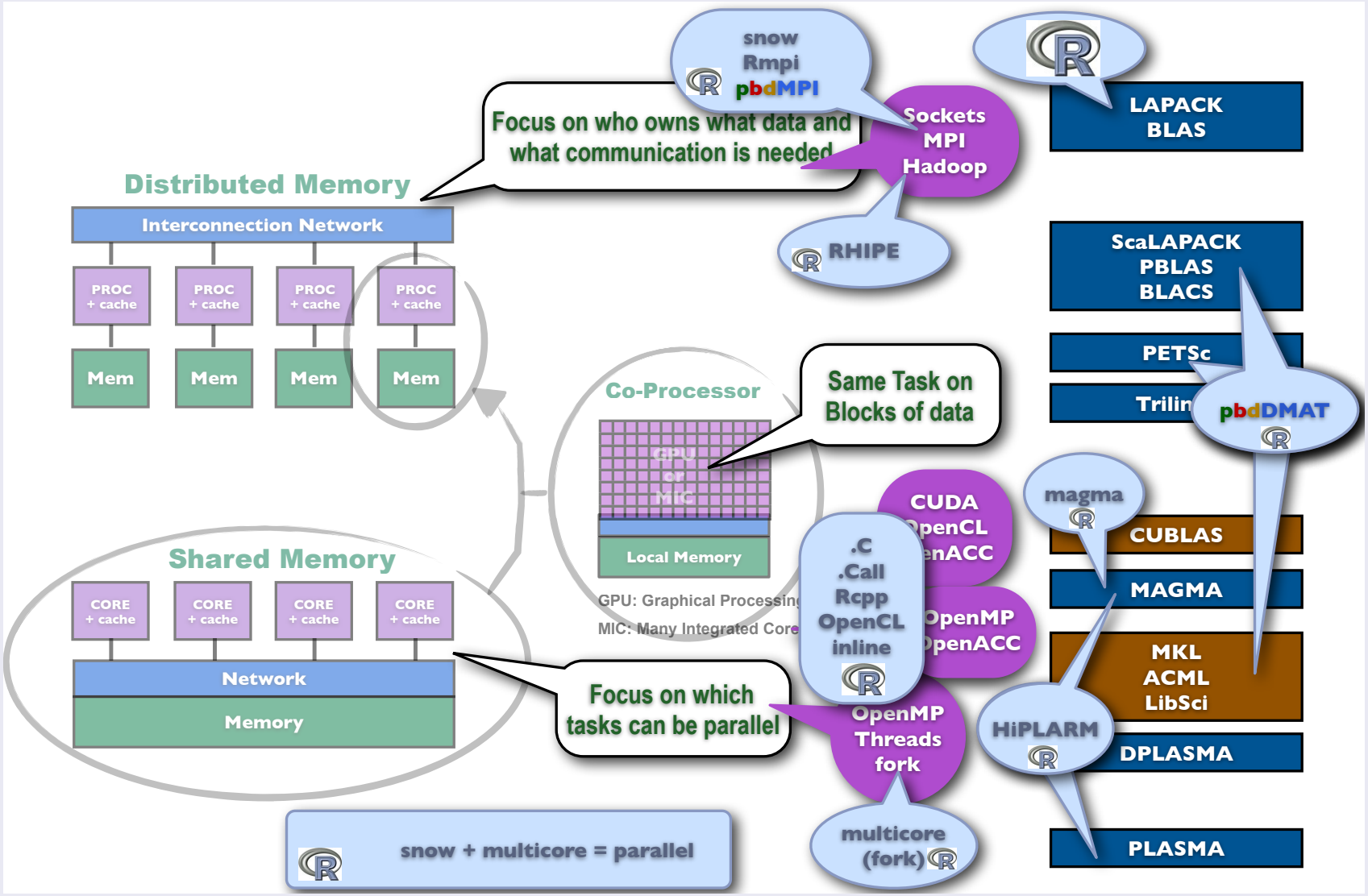
R Interfaces to Native Tools



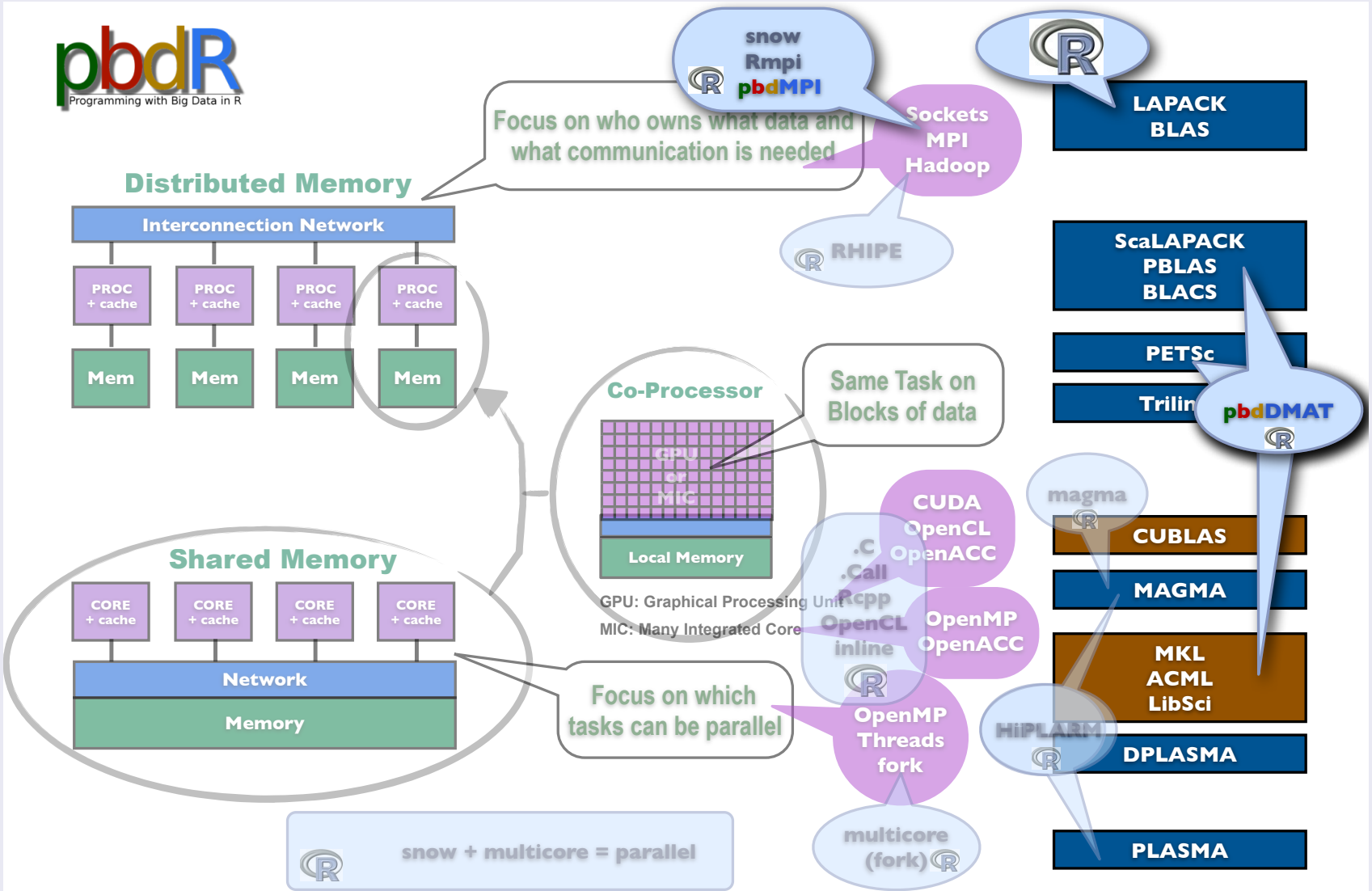
Scalable Math Libraries from 30+ Years of Research



R Interfaces to Scalable Math Libraries



R Interfaces to Scalable Math Libraries



Contents

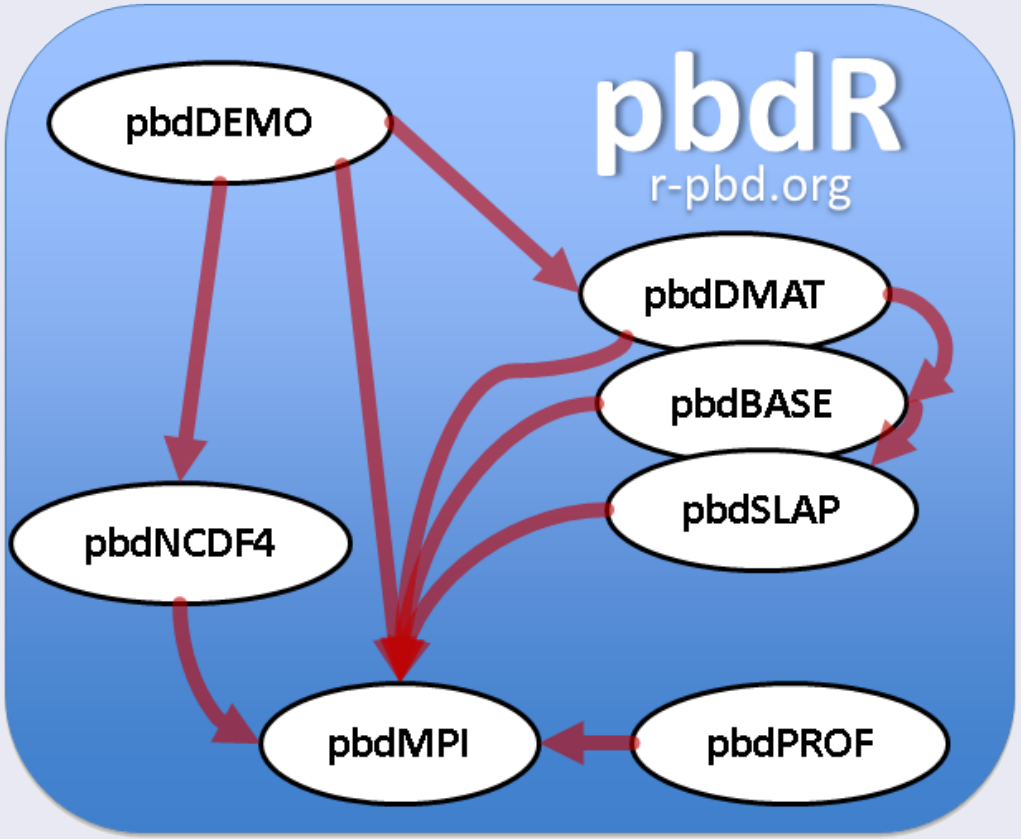
Programming with Big Data in R (pbdR)

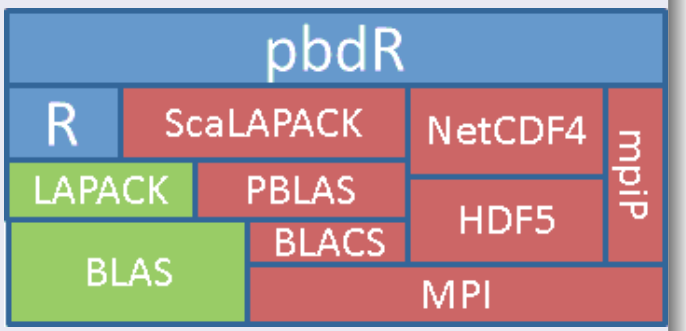
Productivity, Portability, Performance

- *Free*[a] R packages.
- Bridging high-performance C with high-productivity of R
- Distributed data details implicitly managed.
- Methods have syntax *identical* to R.

[a] MPL, BSD, and GPL licensed

pbdR Packages





pbdMPI vs Rmpi: API

Reduction Operations

Rmpi

```
1 # int
2 mpi.allreduce(x, type=1)
3 # double
4 mpi.allreduce(x, type=2)
```

pbdMPI

```
1 allreduce(x)
```

Types in R

```
1 > is.integer(1)
2 [1] FALSE
3 > is.integer(2)
4 [1] FALSE
5 > is.integer(1:2)
6 [1] TRUE
```

MPI Operations: The Gang's All Here

- **Communicator wrangling**: `init()`, `finalize()`
- **Rank query**: `comm.rank()`, `comm.size()`
- **Reduction**: `reduce(x, op='sum')`, `allreduce(x)`
- **Gather**: `gather(x)`, `allgather(x)`
- **Broadcast**: `bcast(x)`
- **Barrier**: `barrier()`
- **Send/Receive**: `send(x)`, `recv()`

MPI Operations for R Users

- **Printing**: `comm.print(x)`, `comm.cat(x)`
- **RNG Seeds**: `comm.set.seed(diff=T)`
- **Task Subsetting**: `get.jid(n)`
- ***ply**:

```
pbdApply(X, MARGIN, FUN, ...)
pbdLapply(X, FUN, ...)
pbdSapply(X, FUN, ...)
```

Rank Query Example

```
1 library(pbdMPI, quiet = TRUE)
2 init()
3
4 my.rank <- comm.rank()
5 comm.print(my.rank, all.rank=TRUE)
6
7 finalize()
```

Execute this script via:

```
1 mpirun -np 2 Rscript 1_rank.r
```

Sample Output:

```
1 COMM.RANK = 0
2 [1] 0
3 COMM.RANK = 1
4 [1] 1
```

2×3 block-cyclic grid on 6 processors: Global view

$$x = \left[\begin{array}{cc|cc|cc|cc|c} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} & x_{28} & x_{29} \\ \hline x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} & x_{38} & x_{39} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} & x_{48} & x_{49} \\ \hline x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} & x_{58} & x_{59} \\ x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} & x_{68} & x_{69} \\ \hline x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77} & x_{78} & x_{79} \\ x_{81} & x_{82} & x_{83} & x_{84} & x_{85} & x_{86} & x_{87} & x_{88} & x_{89} \\ \hline x_{91} & x_{92} & x_{93} & x_{94} & x_{95} & x_{96} & x_{97} & x_{98} & x_{99} \end{array}\right]_{9\times 9}$$

$$\text{Processor grid} = \left|\begin{array}{ccc} 0 & 1 & 2 \\ 3 & 4 & 5 \end{array}\right| = \left|\begin{array}{ccc} (0,0) & (0,1) & (0,2) \\ (1,0) & (1,1) & (1,2) \end{array}\right|$$

2×3 block-cyclic grid on 6 processors: Local view

$$\left[\begin{array}{cc|cc} x_{11} & x_{12} & x_{17} & x_{18} \\ x_{21} & x_{22} & x_{27} & x_{28} \\ \hline x_{51} & x_{52} & x_{57} & x_{58} \\ x_{61} & x_{62} & x_{67} & x_{68} \\ \hline x_{91} & x_{92} & x_{97} & x_{98} \end{array}\right]_{5\times 4} \quad \left[\begin{array}{cc|c} x_{13} & x_{14} & x_{19} \\ x_{23} & x_{24} & x_{29} \\ \hline x_{53} & x_{54} & x_{59} \\ x_{63} & x_{64} & x_{69} \\ \hline x_{93} & x_{94} & x_{99} \end{array}\right]_{5\times 3} \quad \left[\begin{array}{cc} x_{15} & x_{16} \\ x_{25} & x_{26} \\ \hline x_{55} & x_{56} \\ x_{65} & x_{66} \\ \hline x_{95} & x_{96} \end{array}\right]_{5\times 2}$$

$$\left[\begin{array}{cc|cc} x_{31} & x_{32} & x_{37} & x_{38} \\ x_{41} & x_{42} & x_{47} & x_{48} \\ \hline x_{71} & x_{72} & x_{77} & x_{78} \\ x_{81} & x_{82} & x_{87} & x_{88} \end{array}\right]_{4\times 4} \quad \left[\begin{array}{cc|c} x_{33} & x_{34} & x_{39} \\ x_{43} & x_{44} & x_{49} \\ \hline x_{73} & x_{74} & x_{79} \\ x_{83} & x_{84} & x_{89} \end{array}\right]_{4\times 3} \quad \left[\begin{array}{cc} x_{35} & x_{36} \\ x_{45} & x_{46} \\ \hline x_{75} & x_{76} \\ x_{85} & x_{86} \end{array}\right]_{4\times 2}$$

$$\text{Processor grid} = \left|\begin{array}{ccc} 0 & 1 & 2 \\ 3 & 4 & 5 \end{array}\right| = \left|\begin{array}{ccc} (0,0) & (0,1) & (0,2) \\ (1,0) & (1,1) & (1,2) \end{array}\right|$$

pbdR Example Syntax

```
1 x <- x[-1, 2:5]
2 x <- log(abs(x) + 1)
3 xtx <- t(x) %*% x
4 ans <- svd(solve(xtx))
```

The above runs on 1 core with R or 10,000 cores with pbdR

Profiling with **pbdPROF**

1. Rebuild pbdR packages

```
R CMD INSTALL
    pbdMPI_0.2-1.tar.gz \
    --configure-args= \
    "--enable-pbdPROF"
```

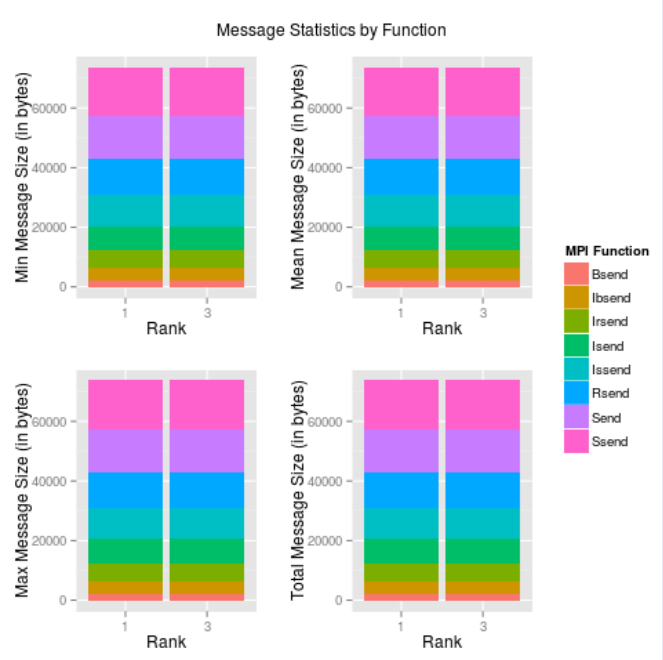
2. Run code

```
mpirun -np 64 Rscript
    my_script.R
```

3. Analyze results

```
1 library(pbdPROF)
2 prof <- read.prof(
      "profiler_output.mpiP")
3 plot(prof)
```

Publication-quality graphs



pbdPAPI

Example

Contents

Non-Optimal Choices Throughout

1. Only free software used (no MKL, ACML, etc.)
2. 1 core = 1 MPI process
3. No tuning for data distribution, just the defaults

Benchmark Data

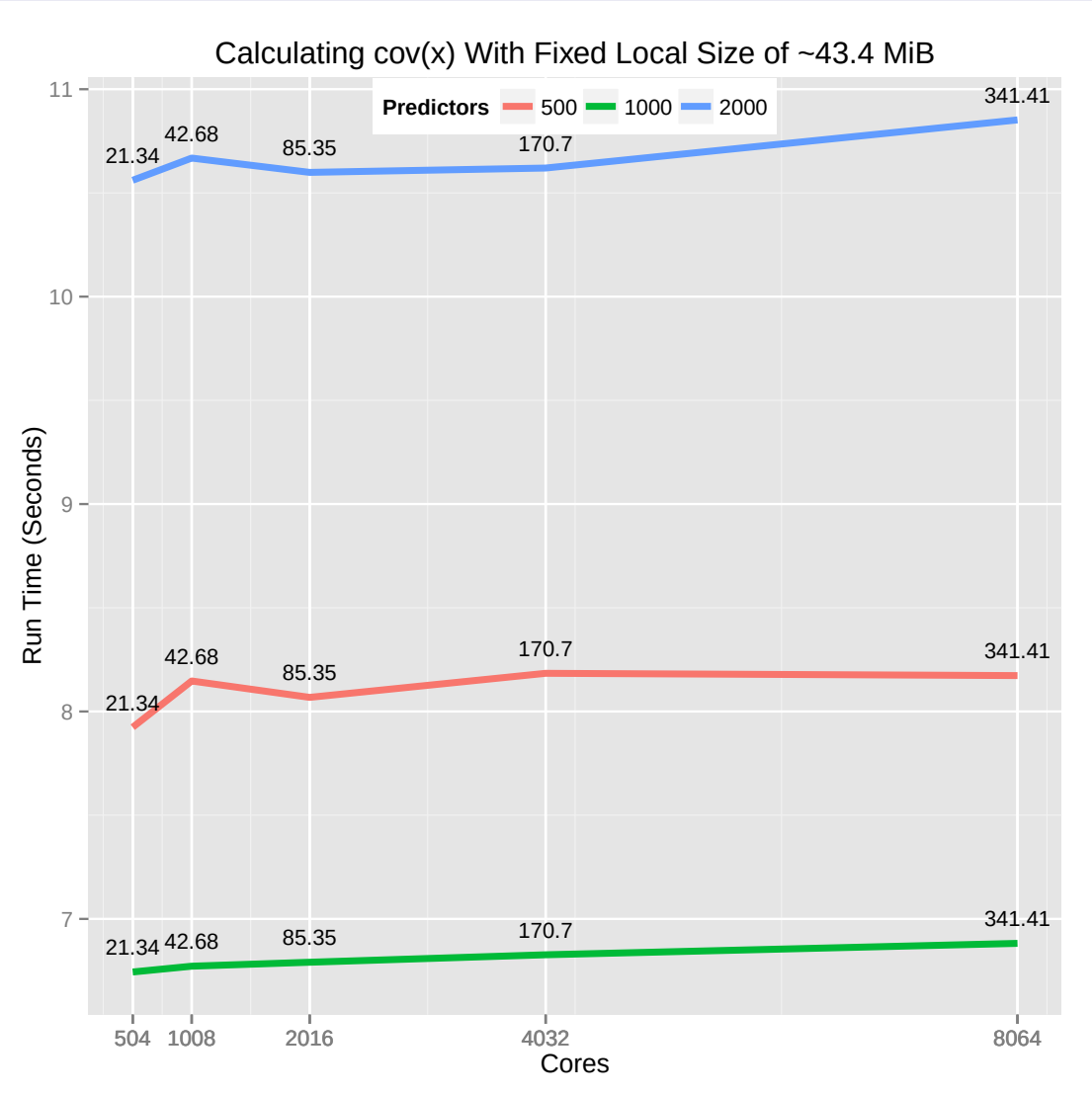
1. Measure wallclock time for covariance and linear regression
2. Random normal $N(100, 10000)$
3. Local problem size fixed at $\approx 43.4 MiB$
4. "weak scaling" = global problem grows with core count
5. Three sets: 500, 1000, and 2000 columns
6. Several runs at different core counts within each set

Covariance Code

$$cov(x_{n\times p}) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)(x_i - \mu_x)^T$$

```
1 x <- ddmatrix("rnorm", nrow=n, ncol=p, mean=mean, sd=sd)
2
3 cov.x <- cov(x)
```

cov()


Calculating cov(x) With Fixed Local Size of ~43.4 MiB
Predictors 500 1000 2000
Run Time (Seconds)
11
10
9
8
7
21.34
42.68
85.35
170.7
341.41
504
1008
2016
4032
8064
Cores

Linear Model Code

Find β such that

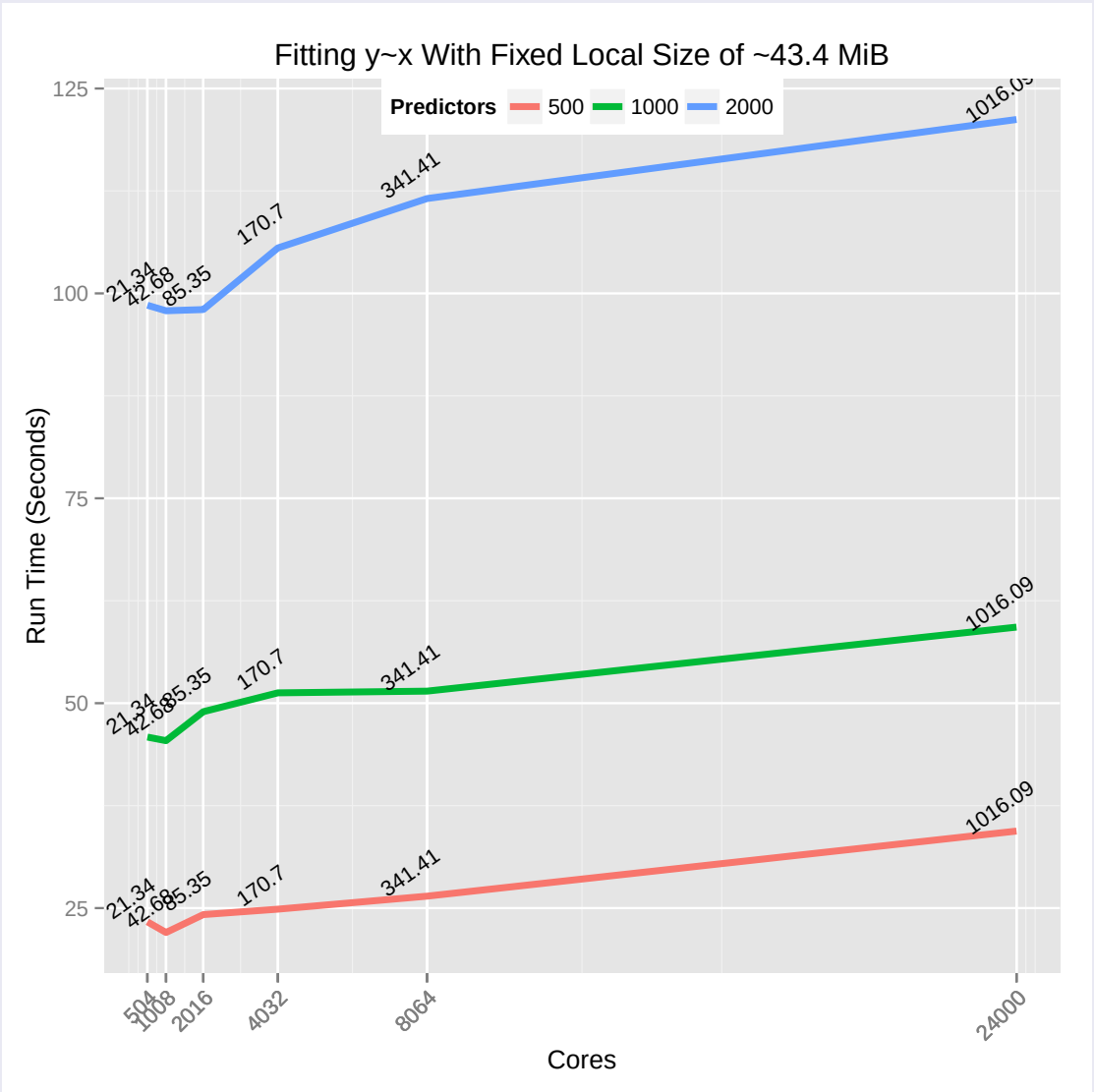
$$y = X\beta + \epsilon$$

When X is full rank,

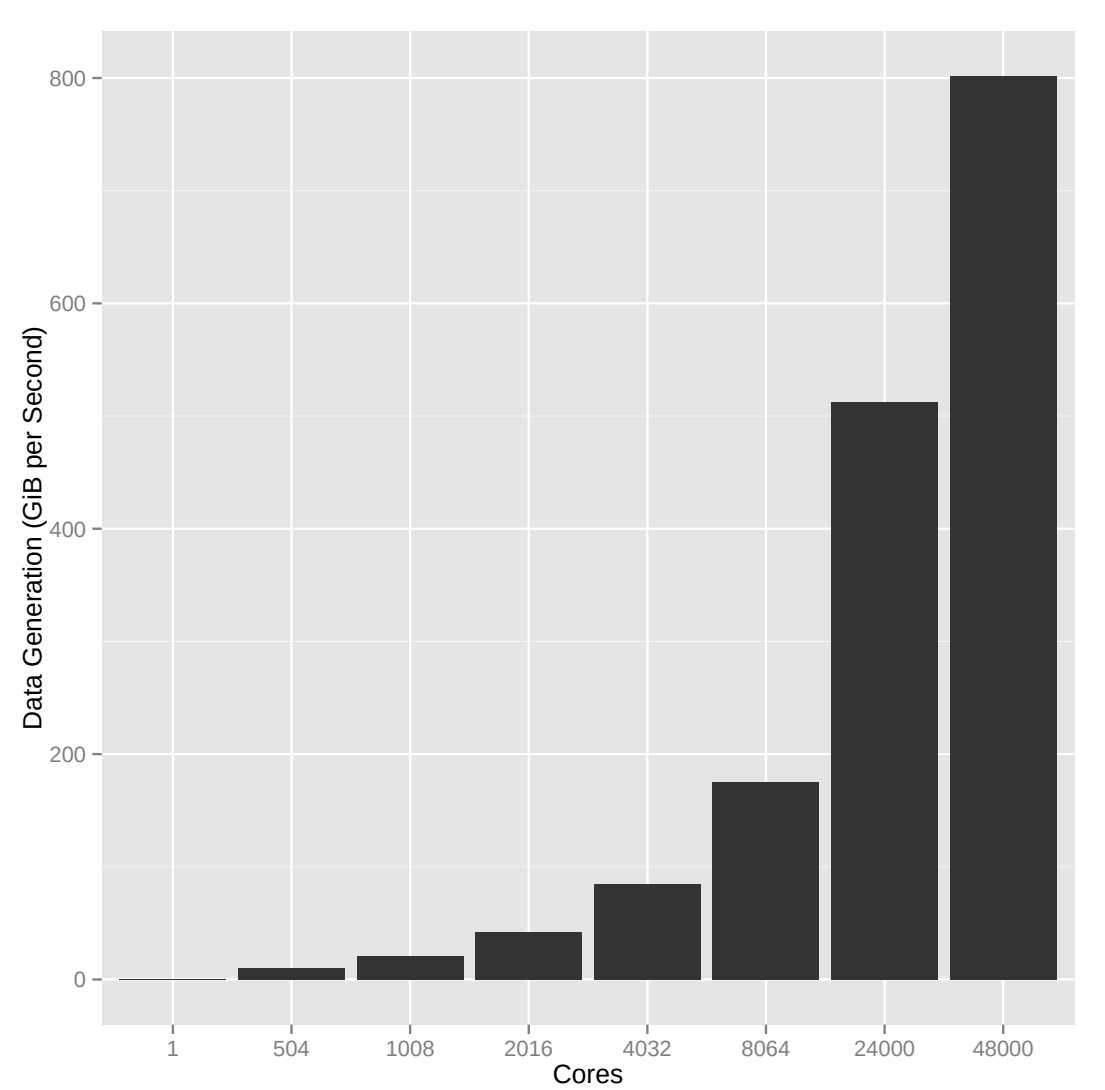
$$\hat{\beta} = (X^T X)^{-1} X^T y$$

```
1 x <- ddmatrix("rnorm", nrow=n, ncol=p, mean=100,
      sd=10000)
2 beta_true <- ddmatrix("runif", nrow=p, ncol=1)
3
4 y <- x %*% beta_true
5
6 beta_est <- lm.fit(x, y)$coefficients
```

lm.fit()


Fitting y~x With Fixed Local Size of ~43.4 MiB
Predictors 500 1000 2000
Run Time (Seconds)
125
100
75
50
25
21.34
42.68
85.35
170.7
341.41
1016.09
504
1008
2016
4032
8064
24000
Cores

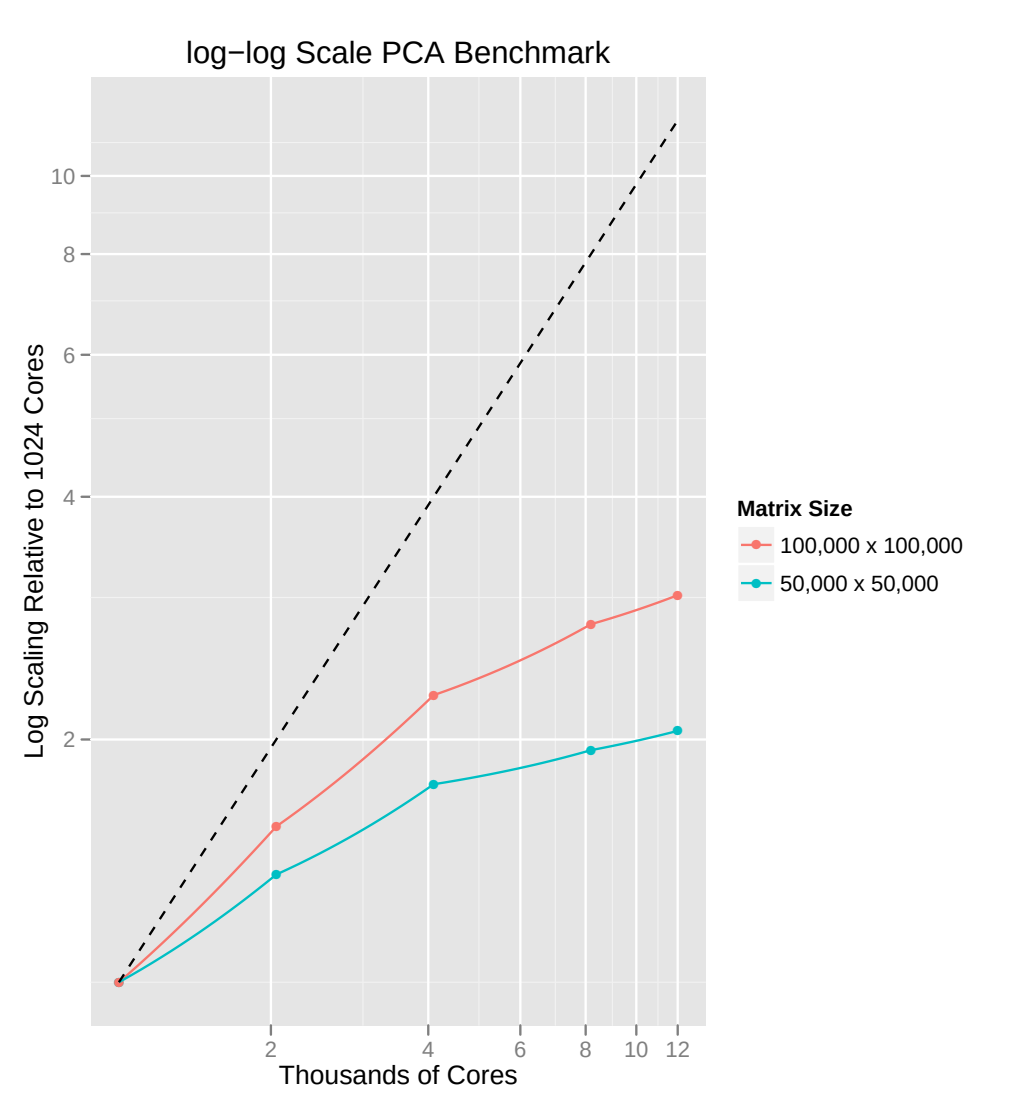
Data Generation



PCA Benchmark Data

1. Measure wallclock time for principal components analysis
2. Random normal $N(100, 10000)$
3. Global problem size fixed
4. "strong scaling" = local problem decreases with core count
5. Two sets: 50,000 $\times$ 50,000 and 100,000 $\times$ 100,000
6. Runs out of local work as core count increases

prcomp() Strong Scaling



Contents

Randomized SVD[1]

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),
5                       nrow=n, ncol=2*k)
6      Y <- A %*% Omega
7      Q <- qr.Q(qr(Y))
8      At <- t(A)
9      for(i in 1:q)
10       {
11         Y <- At %*% Q
12         Q <- qr.Q(qr(Y))
13         Y <- A %*% Q
14         Q <- qr.Q(qr(Y))
15       }
16
17     ## Stage B
18     B <- t(Q) %*% A
19     U <- La.svd(B)$u
20     U <- Q %*% U
21     U[, 1:k]
22   }
```

[1] Halko N, Martinsson P-G and Tropp J A 2011 Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions *SIAM Rev.* **53** 217–88

Randomized SVD

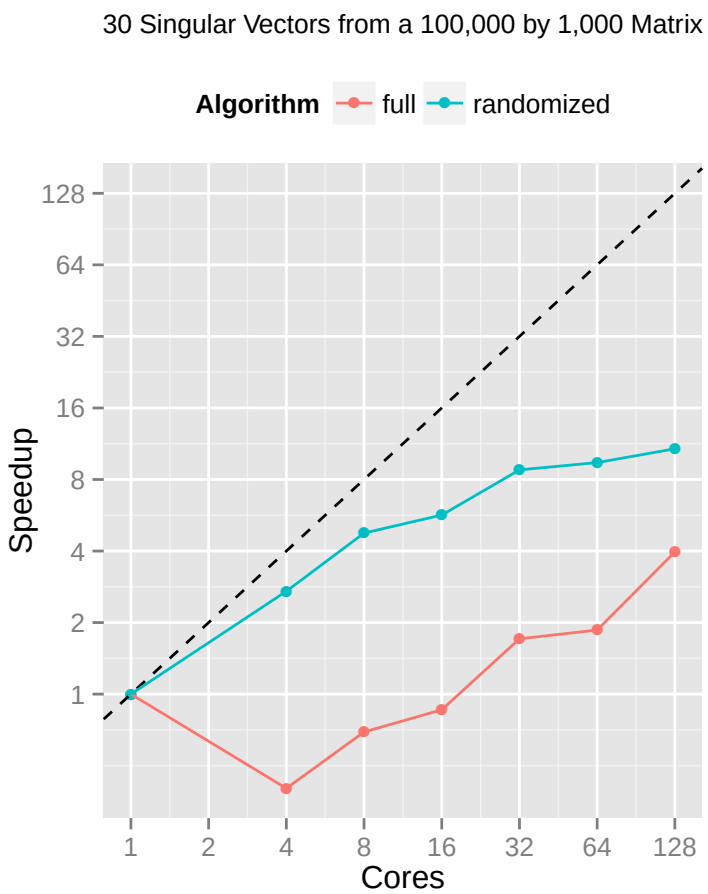
Serial R

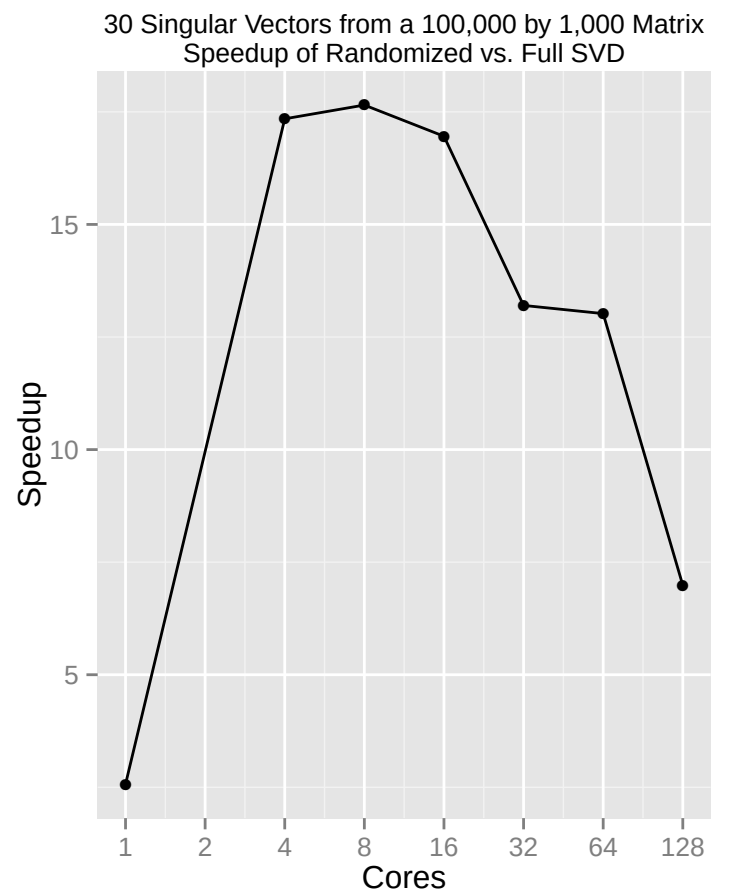
```
1  randSVD <- function(A, k, q=3)
2    {
3      ## Stage A
4      Omega <-  matrix(rnorm(n*2*k),
5              nrow=n, ncol=2*k)
6      Y <- A %*% Omega
7      Q <- qr.Q(qr(Y))
8      At <- t(A)
9      for(i in 1:q)
10       {
11         Y <- At %*% Q
12         Q <- qr.Q(qr(Y))
13         Y <- A %*% Q
14         Q <- qr.Q(qr(Y))
15       }
16
17     ## Stage B
18     B <- t(Q) %*% A
19     U <- La.svd(B)$u
20     U <- Q %*% U
21     U[, 1:k]
22   }
```

Parallel pbdR

```
1  randSVD <- function(A, k, q=3)
2    {
3      ## Stage A
4      Omega <-  ddmatrix("rnorm",
5              nrow=n, ncol=2*k)
6      Y <- A %*% Omega
7      Q <- qr.Q(qr(Y))
8      At <- t(A)
9      for(i in 1:q)
10       {
11         Y <- At %*% Q
12         Q <- qr.Q(qr(Y))
13         Y <- A %*% Q
14         Q <- qr.Q(qr(Y))
15       }
16
17     ## Stage B
18     B <- t(Q) %*% A
19     U <- La.svd(B)$u
20     U <- Q %*% U
21     U[, 1:k]
22   }
```

Randomized SVD on ≈ 765 MiB

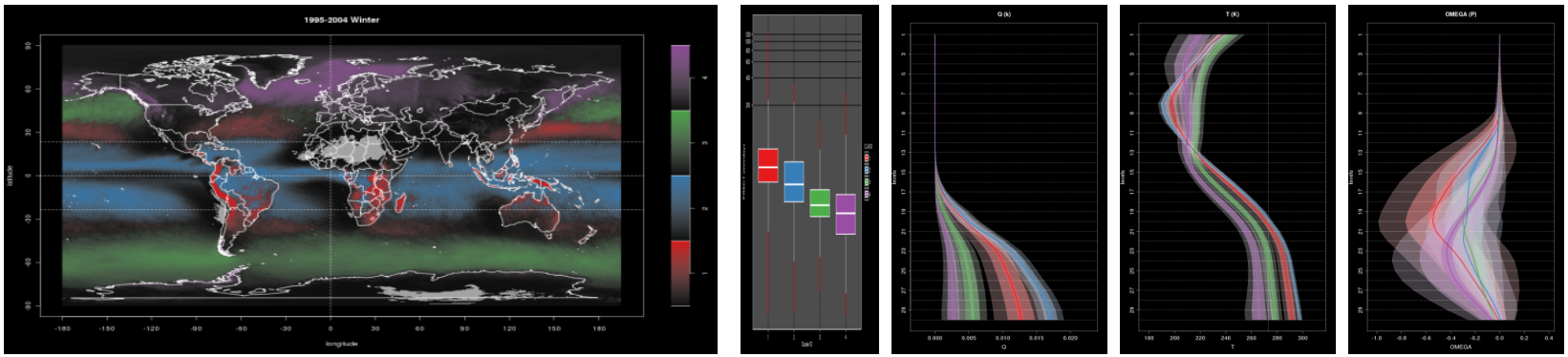




Exploration of Climate Data (2.9 TB) with pbdR

- **2013 INCITE "Attributing Changes in the Risk of Extreme Weather and Climate" Michael Wehner, PI**
  - High resolution atmospheric model, CAM 5.1 (~25 km resolution)
  - Resolves processes responsible for extreme precipitation and storms
- **R and pbdR used for scalable analytics and graphics**
  - Advanced clustering capability (EM algorithm for Gaussian mixture models)

Behavior of extreme precipitation across atmospheric layers of moisture, temperature, and wind



W.-C. Chen, G. Ostrouchov, D. Pugmire, Prabhat, M. Wehner. A Parallel EM Algorithm for Model-Based Clustering Applied to the Exploration of Large Spatio-Temporal Data. *Technometrics* (online, in print: Fall 2013).

Simulation in the Social Sciences with pbdR

- Survey responses can be intentionally misleading
- This results in bias and poor policy decisions
- Simulation examines bias

"... [Serial] simulations would require **nearly a year** *to complete. ... Using [pbdR] and NICS resources,* **the simulation completed in just 15 hours**."

— Joseph Robinson, Educational Psychology, University of Illinois at Urbana-Champaign

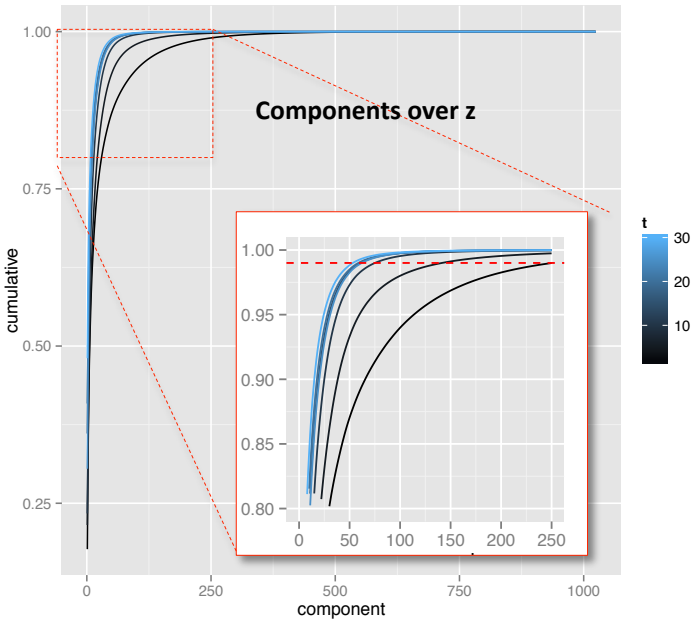
Engaging with the R Community

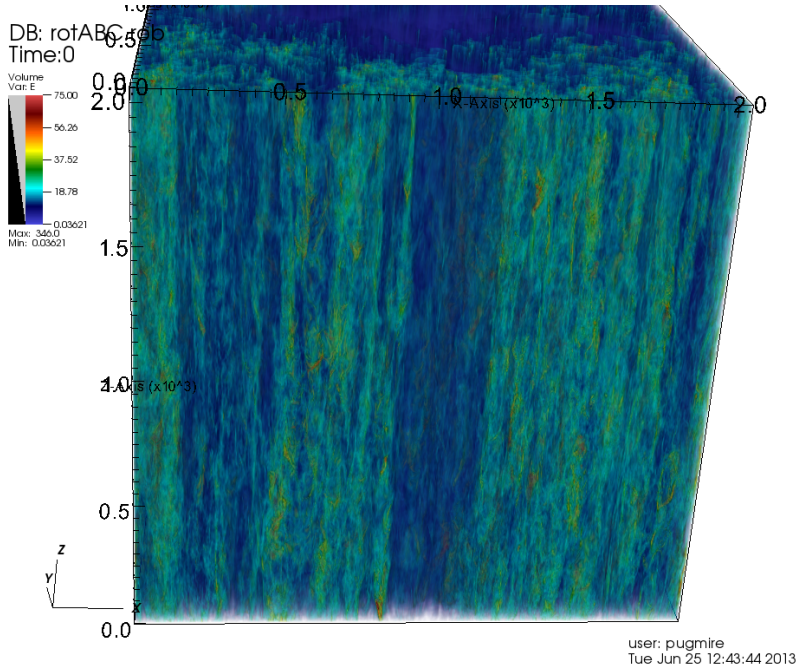
"It's exactly what I wanted ever since I started learning about HPC with C and MPI a few years back!"

— Andrew Raim, Department of Mathematics and Statistics, University of Maryland, Baltimore County

Coherent Structures in Turbulent Flow with pbdR

- Data: Turbulence cube: $2048^3 \approx 8.6$ Billion elements $\approx$ 70 GiB per time step
- Principal components analysis across z
- Subset: 512 x 512 x 1,024
- I/O + subset + PCA + plots $\approx$ 60 seconds per time step with 128 cores





- **Coherence in z increases over time:**
- *Time 1:* **25%** components capture 99% variability
- *Time 30:* **5%** components capture 99% variability

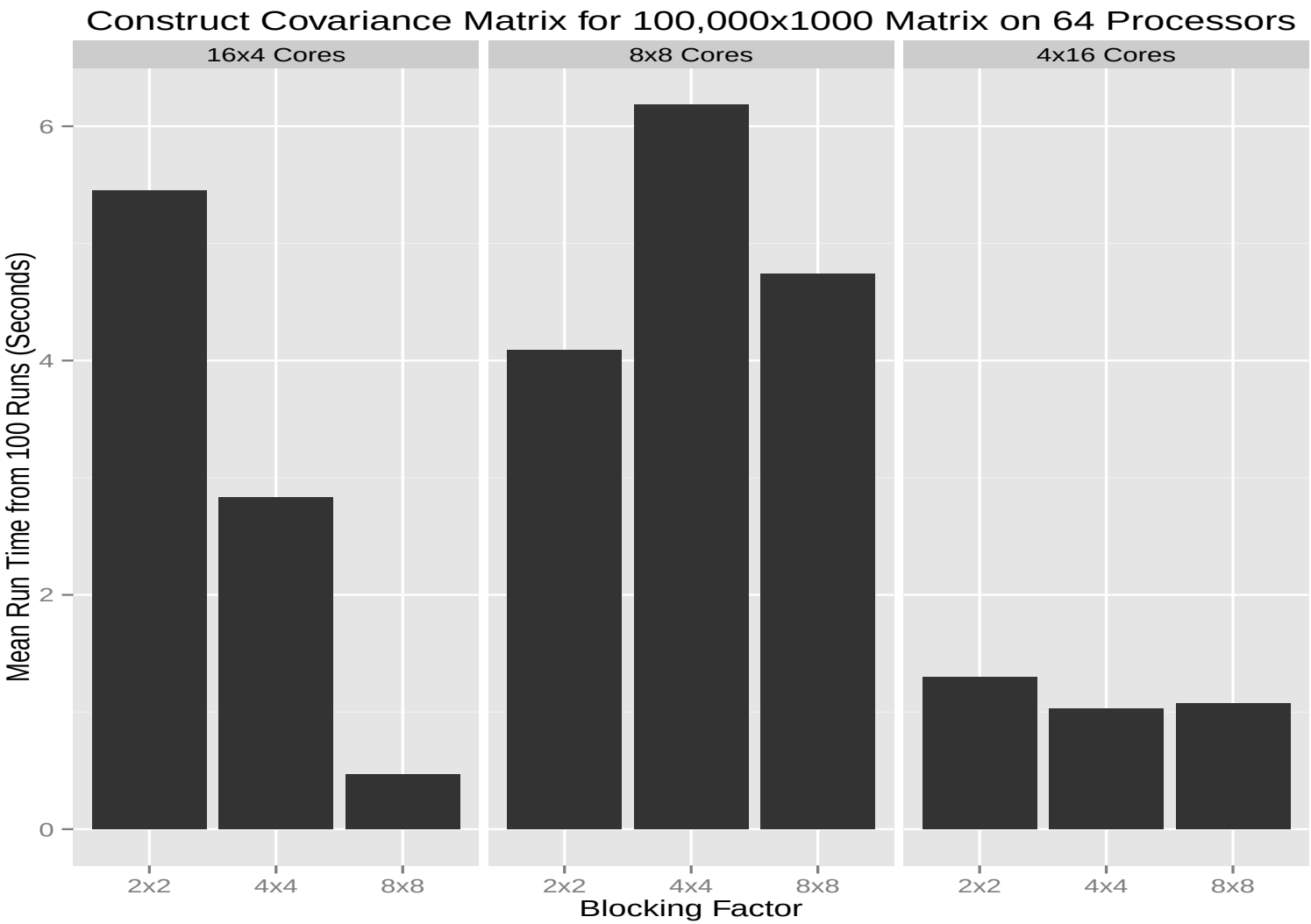
2013 INCITE "Parameter Studies of Boussinesq Flows" Susan Kurien, PI, Duane Rosenberg, Co-PI, et al

Contents

Challenges

- Perceptions.
  "*R? Isn't that slow?*" – HPC people
  "*HPC? Isn't that hard?*" – R people
- Bringing R community to large platforms
- Distributed data input
- Bringing interactivity back
- Performance dependence on data layout

Covariance Revisited: Data layout parameter calibration



Future Work

- Starting a 3 year NSF grant to work on
  - Bring back interactivity via client/server
  - Simplify parallel data input
  - Begin DPLASMA integration
- Optimization for heterogeneous architectures
- Support for more profilers
- In-situ use with simulations (QMC, AutoTune)

Where to learn more?

- http://r-pbd.org/
- **pbdDEMO** vignette
- RBigData@gmail.com

Thanks for coming!

Questions?

http://r-pbd.org/