From 1 core to Thousands: R to pbdR

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

[1]Oak Ridge National Laboratory, Oak Ridge, TN
[2]University of Tennessee, Knoxville, TN

OLCF Workshop
on Processing and Analysis of Very Large Data Sets
August 8, 2013

The pbdR Core Team

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

299,008 Cores and 18,688 GPUs in 18,688 Nodes with 762 TB Memory

Support

This work 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 work also 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 used resources of the Newton HPC Program at the University of Tennessee, Knoxville.

[1]Oak Ridge National Laboratory. Supported in part by the project "Visual Data Exploration and Analysis of Ultra-large Climate Data" funded by U.S. DOE Office of Science under Contract No. DE-AC05-00OR22725.

[2]University of Tennessee. Supported in part by the project "NICS Remote Data Analysis and Visualization Center" funded by the Office of Cyberinfrastructure of the U.S. National Science Foundation under Award No. ARRA-NSF-OCI-0906324 for NICS-RDAV center.

Contents

Contents

What is R?

- *lingua franca* for data analytics and statistical computing.
- Part programming language, part data analysis package.
- Dialect of S (Bell Labs).
- Syntax designed for data, scoping semantics, and 2 official OOP systems.

Who uses R?

Google, Pfizer, Merck, Bank of America, Shell[a], Oracle[b], Facebook, bing, Mozilla, okcupid[c], ebay[d], kickstarter[e], the New York Times[f]

---

[a] https://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=0

[b] http://www.oracle.com/us/corporate/features/features-oracle-r-enterprise-498732.html

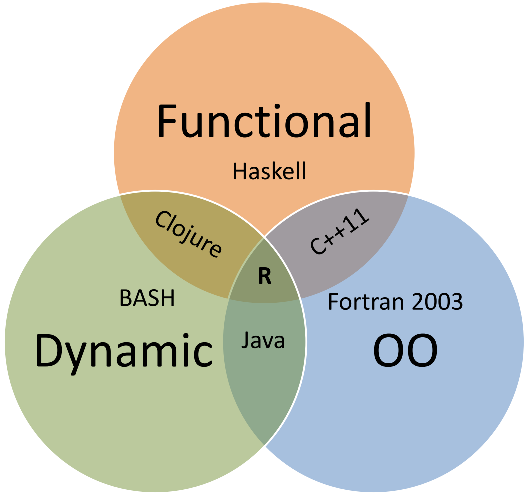
[c] http://www.revolutionanalytics.com/what-is-open-source-r/companies-using-r.php

[d] http://blog.revolutionanalytics.com/2012/09/using-r-in-production-industry-experts-share-their-experiences.html

[e] http://blog.revolutionanalytics.com/2012/09/kickstarter-facilitates-50m-in-indie-game-funding.html

[f] http://blog.revolutionanalytics.com/2012/05/nyt-charts-the-facebook-ipo-with-r.html

Language Paradigms



Data Types

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

High Level Syntax

```
x <- matrix(rnorm(30), nrow=10)
x <- x[-1, 2:5]
x <- log(abs(x) + 1)
xtx <- t(x) %*% x
ans <- svd(solve(xtx))
```

More than just a Matlab clone...

- Data science (machine learning, statistics, data mining, ...) is mostly matrix algebra.

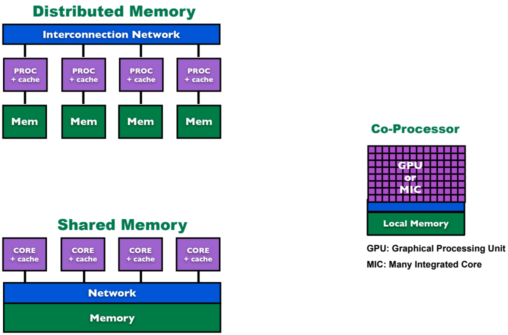
  So what about Matlab/Python/Julia/...?
- Depends on your "religion"
- As a *data analysis* package, R is king.

High Level Syntax *for Data*

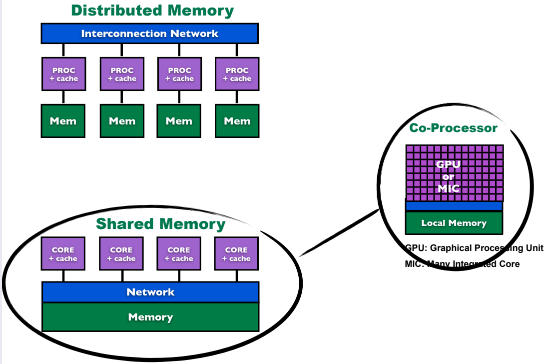
```r
pca <- prcomp(x, retx=TRUE, scale=TRUE)
prop_var <- cumsum(pca$sdev)/sum(pca$sdev)
i <- min(which(prop_var > 0.9)) - 1

y <- pca$x[, 1:i]
```

Contents

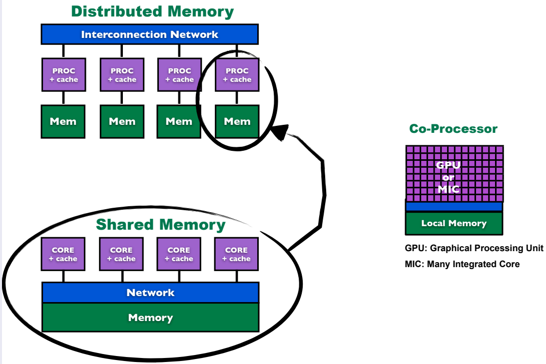
Three Basic Flavors of Hardware



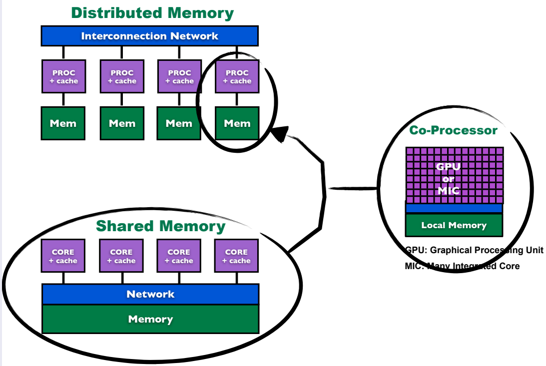
Your Laptop or Desktop



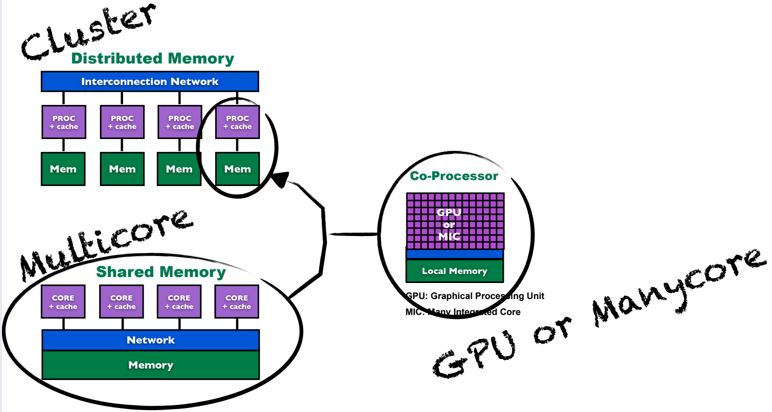
A Server or Cluster



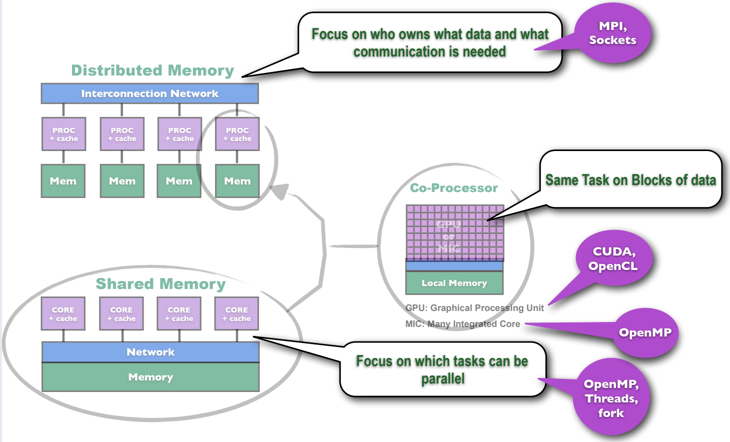
Server to Supercomputer



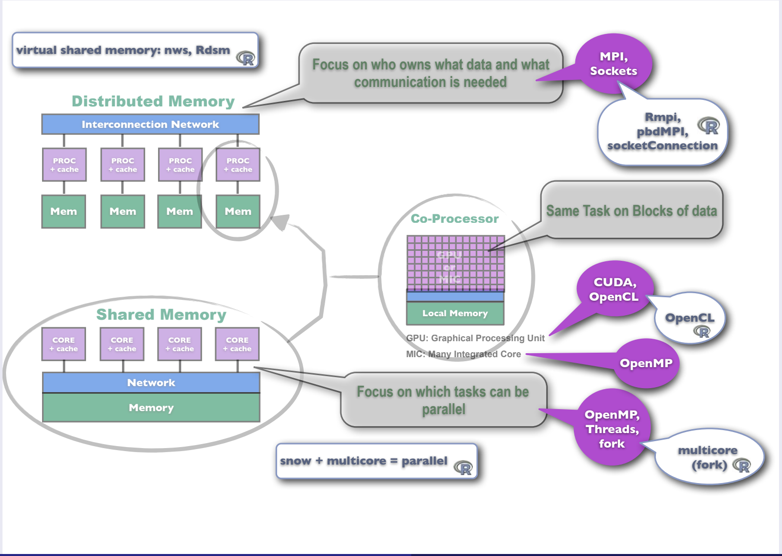
Knowing the Right Words



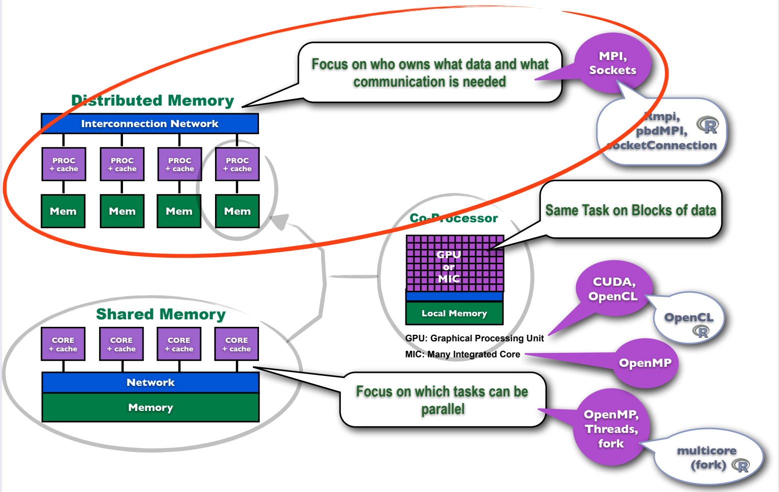
"Native" Programming Models and Tools



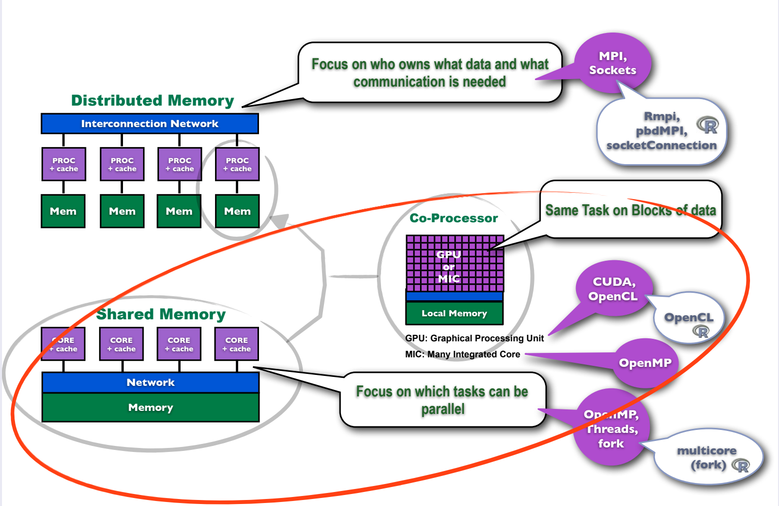
R Interfaces to Native Tools



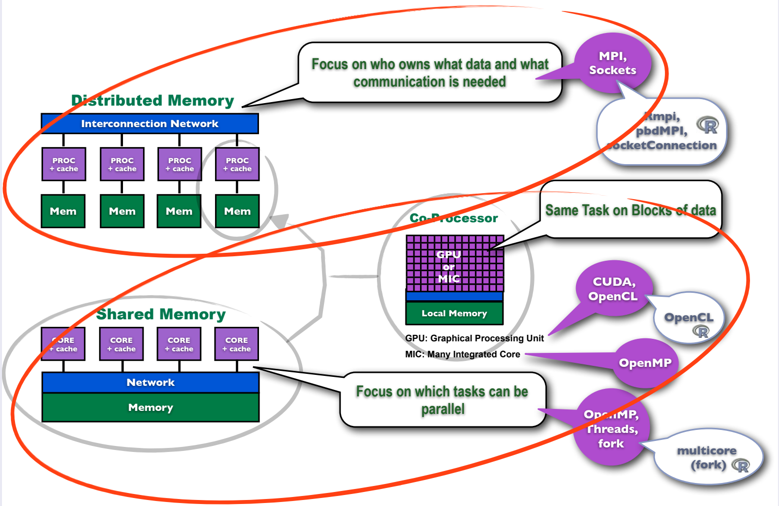
30+ Years of Parallel Computing Research



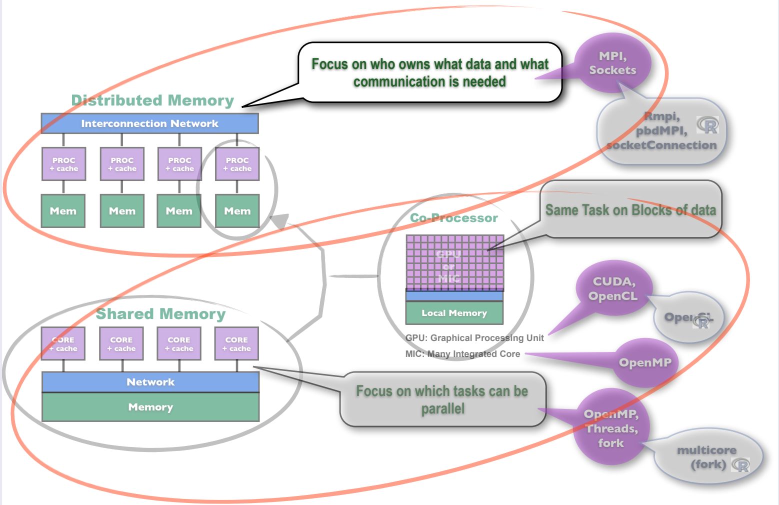
Last 10 years of Advances



Putting It All Together Challenge



pbdR: Focus on Data Parallelism



Contents

Programming with Big Data in R (pbdR)

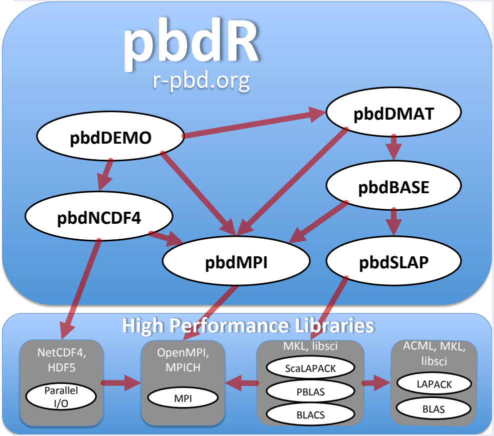
Productivity, Portability, Performance

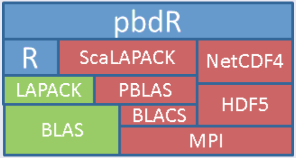
pbdR
Programming with Big Data in R

- *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





pbdR on HPC Resources

pbdR is currently installed and maintained on:

- Nautilus, UTK
- Kraken, UTK
- Newton, UTK
- Lens, ORNL
- Titan, ORNL
- tara, UMBC

If you are interested in maintaining pbdR, contact us at RBigData@gmail.com

pbdR Example Syntax

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

Look familiar?

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

It also runs on 2 cores or 4 cores of your laptop with pbdR

pbdR Paradigms

Programs that use pbdR utilize:

- Batch execution
- Single Program/Multiple Data (SPMD) style

And generally utilize:

- Data Parallelism

Batch Execution

- Non-interactive
- Use

```
Rscript my_script.r
```

or

```
R CMD BATCH my_script.r
```

- In parallel:

```
mpirun -np 2 Rscript my_par_script.r
```

Single Program/Multiple Data (SPMD)

- Difficult to describe, easy to do. . .
- Only one program is written, executed in batch on all processors.
- Different processors are autonomous; there is no manager.
- The dominant programming model for large machines.

Kraken: 112,896 cores in 9,408 nodes with 147 TB Memory

Contents

Truncated SVD from Random Projections[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: $\boldsymbol{B} = \widetilde{\boldsymbol{U}}\boldsymbol{\Sigma V}^*$.
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

```r
randSVD <- function(A, k, q=3)
  {
    ## Stage A
    Omega <- matrix(rnorm(n*2*k),
                    nrow=n, ncol=2*k)
    Y <- A %*% Omega
    Q <- qr.Q(qr(Y))
    At <- t(A)
    for(i in 1:q)
      {
        Y <- At %*% Q
        Q <- qr.Q(qr(Y))
        Y <- A %*% Q
        Q <- qr.Q(qr(Y))
      }

    ## Stage B
    B <- t(Q) %*% A
    U <- La.svd(B)$u
    U <- Q %*% U
    U[, 1:k]
  }
```

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

Randomized SVD

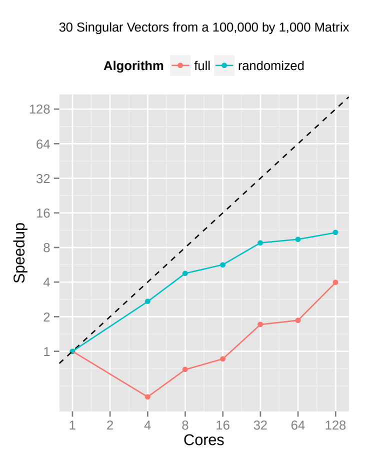
Serial R

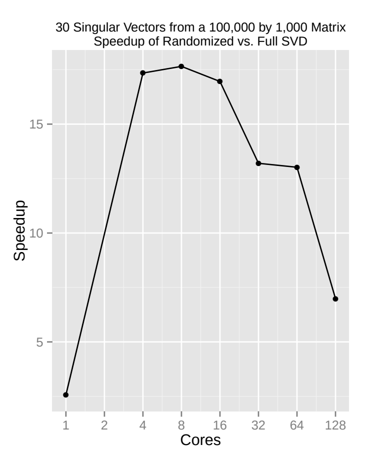
```
randSVD <- function(A, k, q=3)
  {
    ## Stage A
    Omega <- matrix(rnorm(n*2*k),
               nrow=n, ncol=2*k)
    Y <- A %*% Omega
    Q <- qr.Q(qr(Y))
    At <- t(A)
    for(i in 1:q)
      {
        Y <- At %*% Q
        Q <- qr.Q(qr(Y))
        Y <- A %*% Q
        Q <- qr.Q(qr(Y))
      }

    ## Stage B
    B <- t(Q) %*% A
    U <- La.svd(B)$u
    U <- Q %*% U
    U[, 1:k]
  }
```

Parallel pbdR

```
randSVD <- function(A, k, q=3)
  {
    ## Stage A
    Omega <- ddmatrix("rnorm",
               nrow=n, ncol=2*k)
    Y <- A %*% Omega
    Q <- qr.Q(qr(Y))
    At <- t(A)
    for(i in 1:q)
      {
        Y <- At %*% Q
        Q <- qr.Q(qr(Y))
        Y <- A %*% Q
        Q <- qr.Q(qr(Y))
      }

    ## Stage B
    B <- t(Q) %*% A
    U <- La.svd(B)$u
    U <- Q %*% U
    U[, 1:k]
  }
```

Randomized SVD





Non-Optimal Choices Throughout

1. Only libre software used (no MKL, ACML, etc.).
2. 1 core = 1 MPI process.
3. No tuning for data layout.

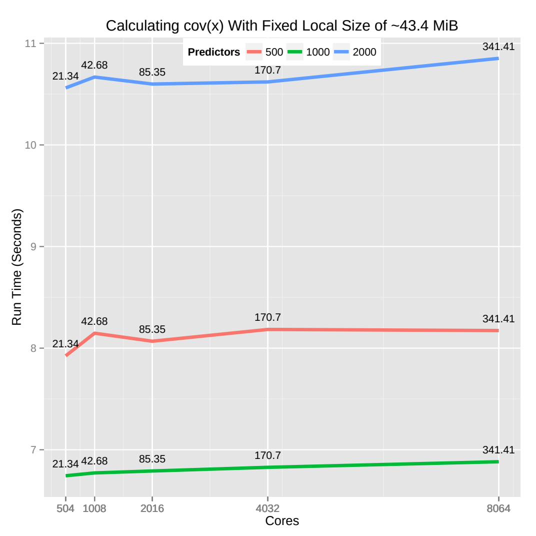
Benchmark Data

1. Random normal $N(100, 10000)$.
2. Local problem size of $\approx 45.5MB$.
3. Three sets: 500, 1000, and 2000 columns.
4. Several runs at different core sizes within each set.

Covariance Code

```r
x <- ddmatrix("rnorm", nrow=n, ncol=p, mean=mean, sd=sd)

cov.x <- cov(x)
```

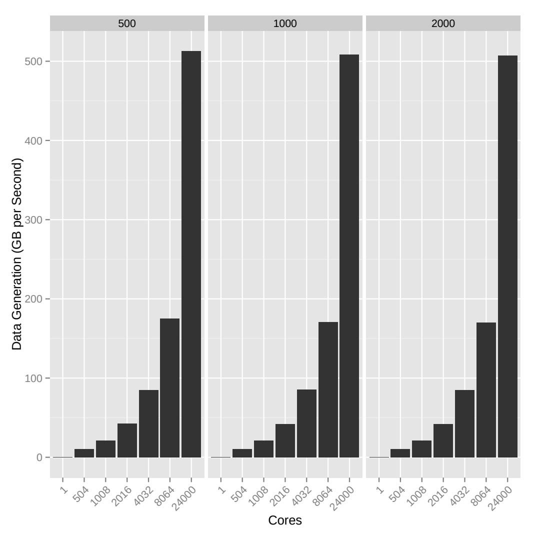
cov()


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

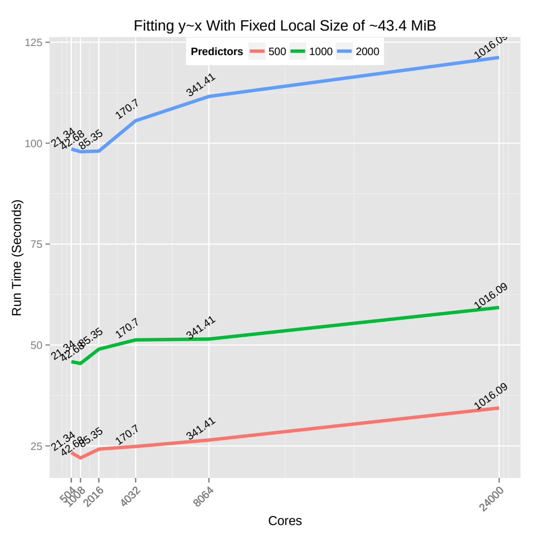
Linear Model Code

```r
x <- ddmatrix("rnorm", nrow=n, ncol=p, mean=mean, sd=sd)
beta_true <- ddmatrix("runif", nrow=p, ncol=1)

y <- x %*% beta_true

beta_est <- lm.fit(x=x, y=y)$coefficients
```

Data Generation



lm.fit()

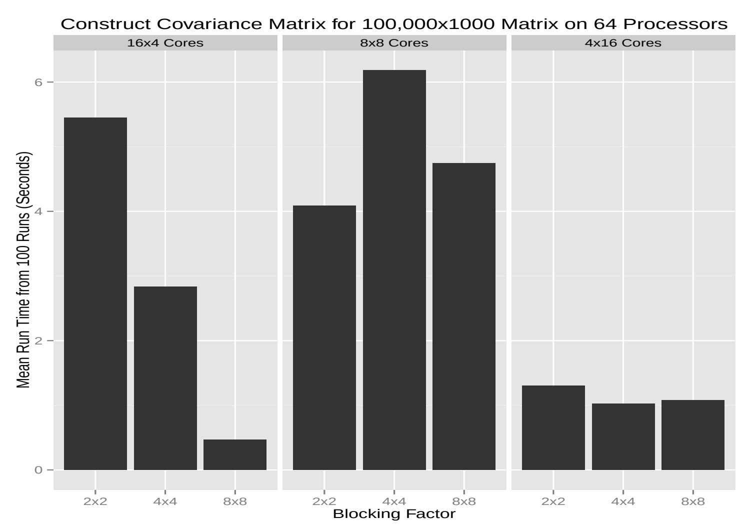

Fitting y~x With Fixed Local Size of ~43.4 MiB
Predictors
500
1000
2000
Run Time (Seconds)
125
100
75
50
25
Cores
504
1008
2016
4032
8064
24000
21.34
42.68
85.35
170.7
341.41
1016.09

Contents

Challenges

- Perceptions.
- Library loading.
- Profiling.

Covariance Revisited: Distributed Data Parameter Calibration



Adding More Levels of Parallelism

Distributed Memory (cluster nodes)
Shared Memory (multicore)
Co-Processor (GPU, manycore)

- pbdDMAT + CUBLAS: near term on Titan
- pbdDMAT - ScaLAPACK + DPLASMA: QR only
- pbdDMAT + PLASMA or MKL or ACML: often helps
- pbdDMAT + MAGMA: may not help

Tutorials

- OLCF Very Large Data Workshop ... NEXT!
- Seoul National University, August 20
- SC13, November 17-22, Denver

Invited Talks

- International Association for Statistical Computing, Aug 22-23, Seoul
- 59th ISI World Statistics Congress, August 25-30, Hong Kong

Thanks for coming!

Questions?

Be sure to stick around for the tutorial!