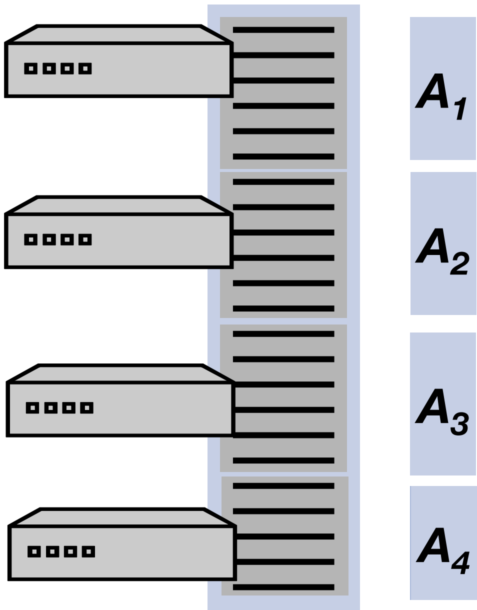



# Tall-and-skinny QRs and SVDs in MapReduce

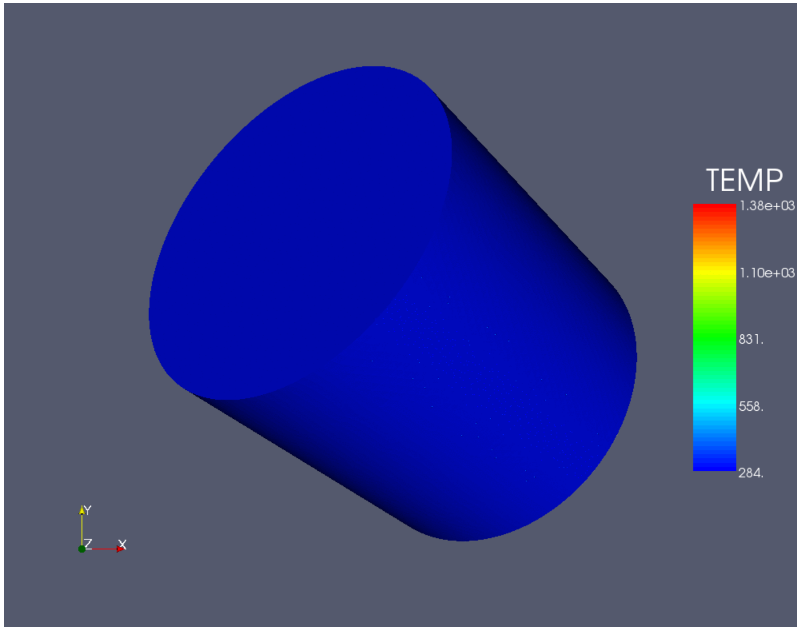
Yangyang Hou
Purdue, CS
Austin Benson
Stanford University
Paul G. Constantine
Col. School. Mines
Joe Nichols – U. of Minn
James Demmel
UC Berkeley
Joe Ruthruff
Jeremy Templeton
Sandia CA

Mainly funded by Sandia National Labs CSAR project, recently by NSF CAREER, and Purdue Research Foundation.

David F. Gleich

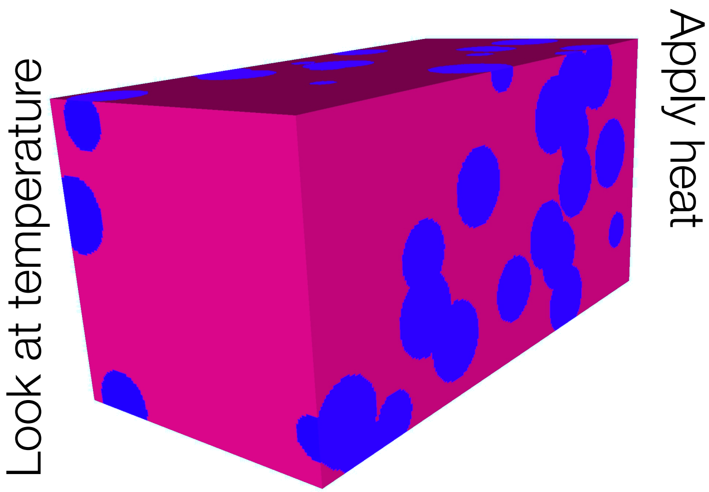
Computer Science
Purdue University

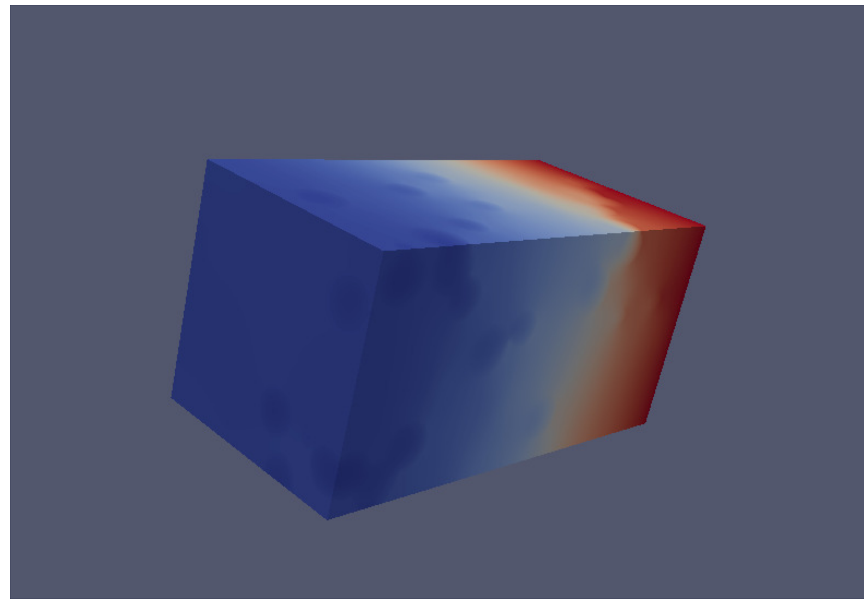
# Big simulation data



# Nonlinear heat transfer model in random media

Each run takes 5 hours on 8 processors, outputs 4M (node) by 9 (time-step) simulation

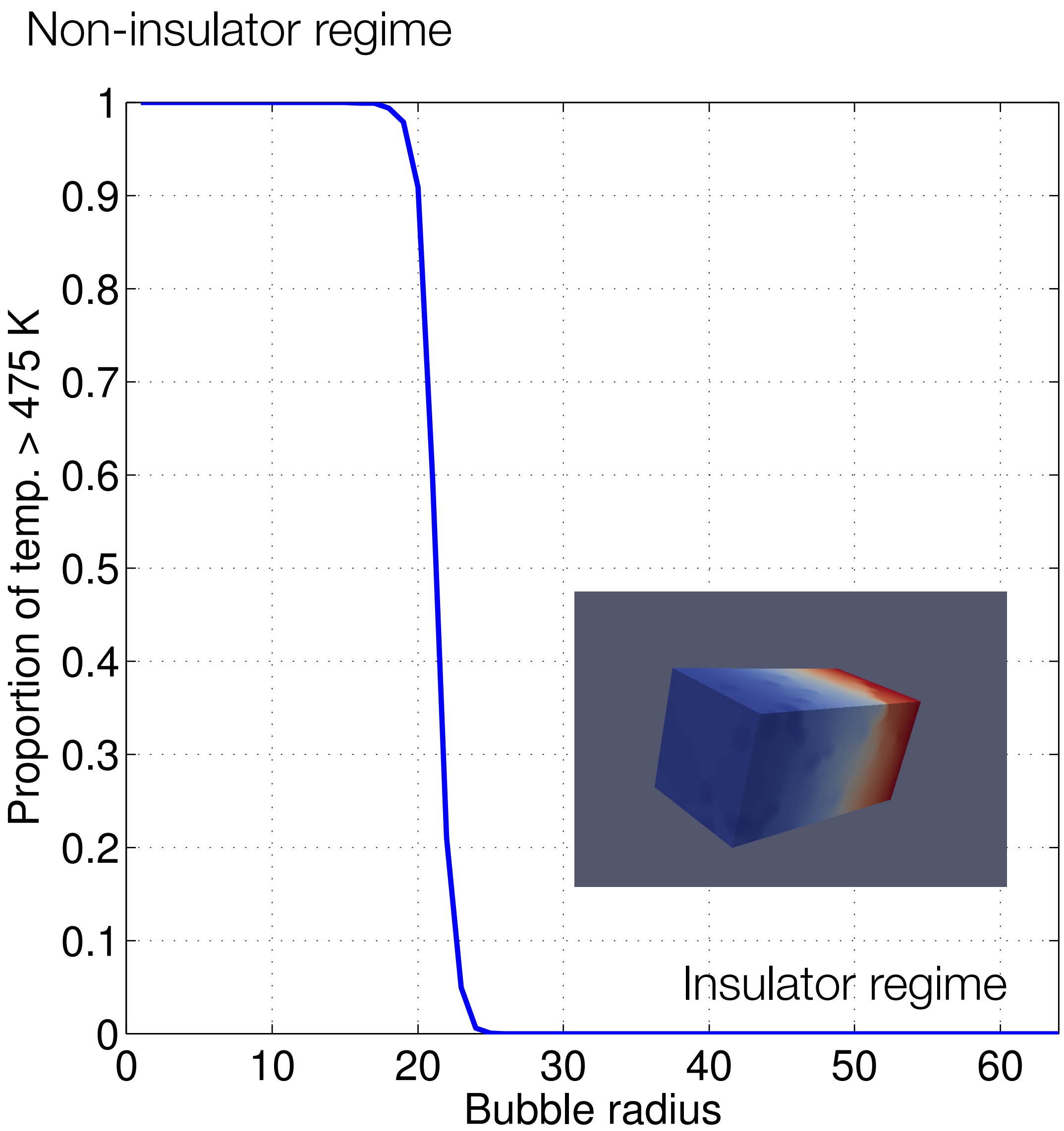


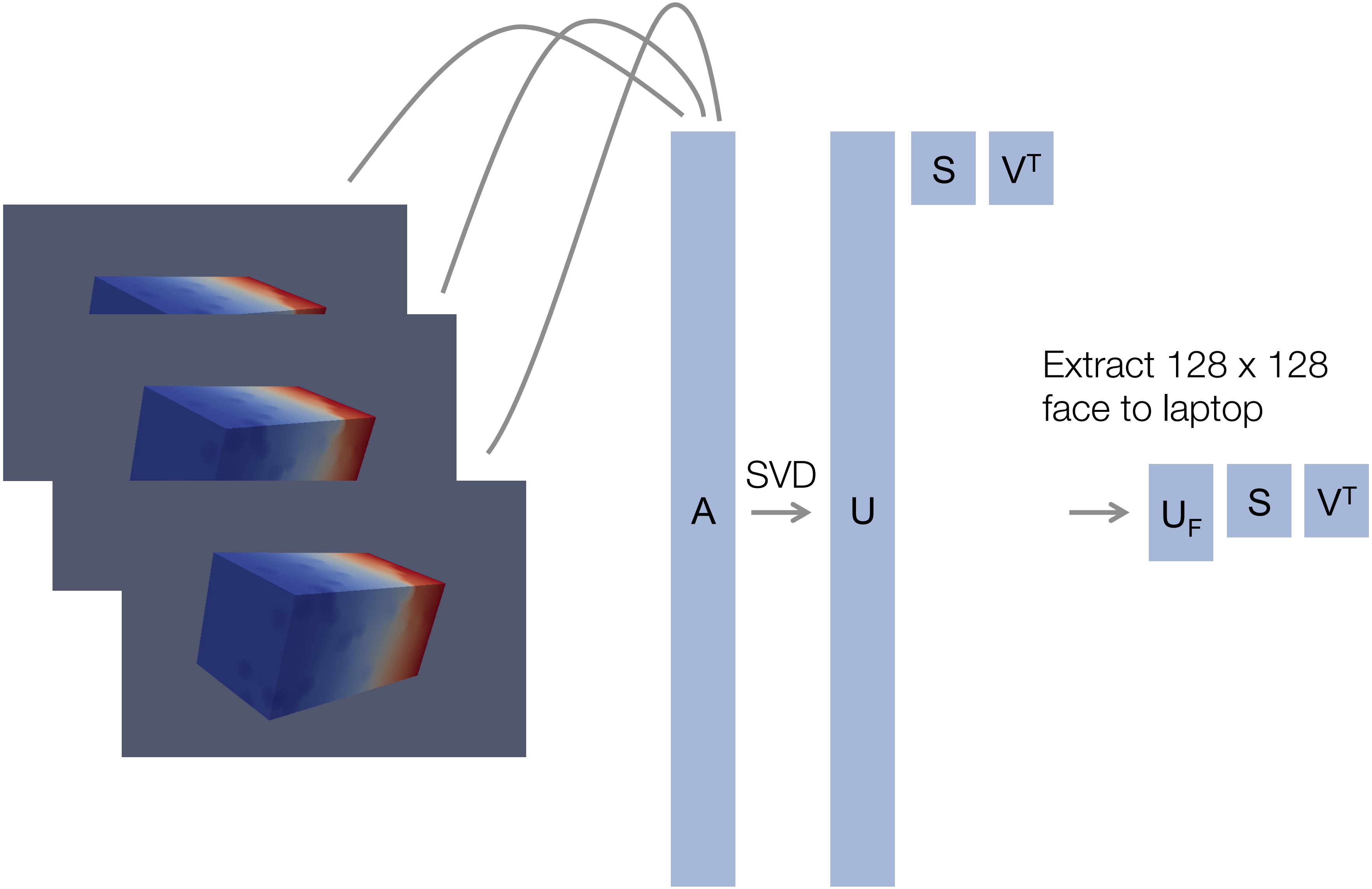



We did 8192 runs (128 samples of bubble locations, 64 bubble radii)

4.5 TB of data in Exodus II (NetCDF)

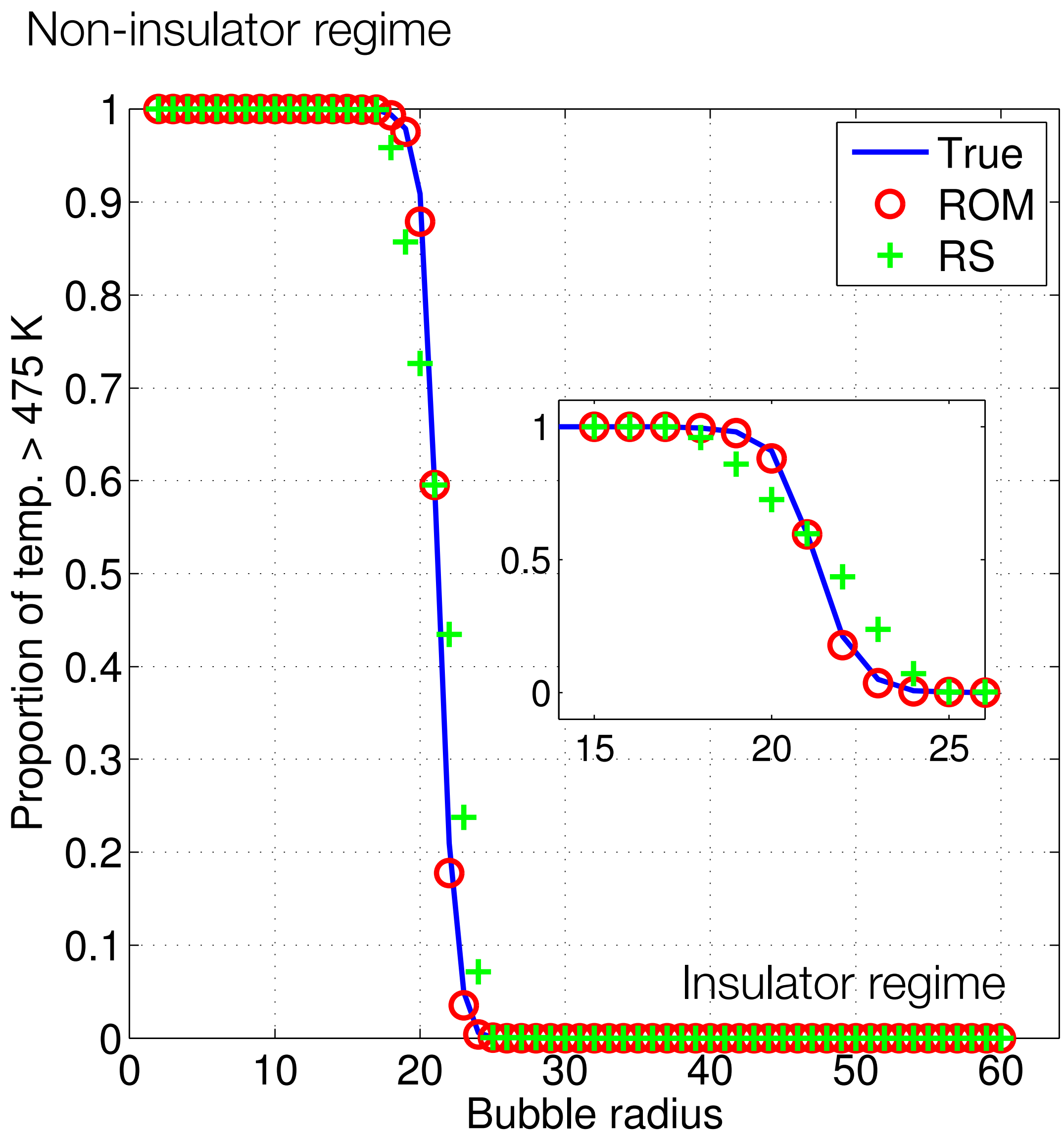
https://www.opensciencedatacloud.org/publicdata/heat-transfer/

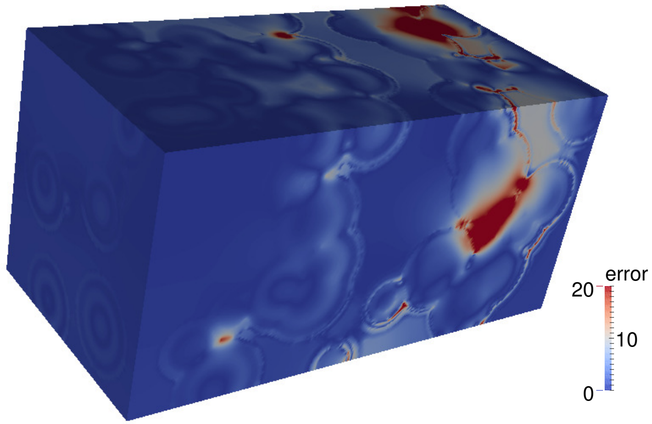

Non-insulator regime
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Proportion of temp. > 475 K
0
10
20
30
40
50
60
Bubble radius
Insulator regime



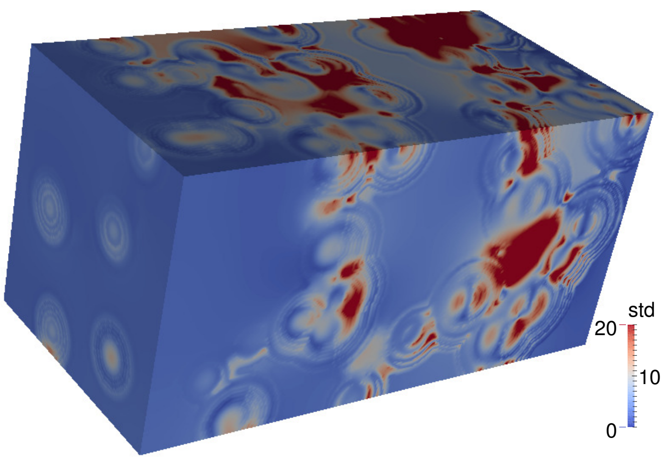

Each simulation is a column

5B-by-64 matrix
2.2TB


Non-insulator regime
Proportion of temp. > 475 K
Bubble radius
True
ROM
RS
Insulator regime
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
1
0.5
0
15
20
25




(c) Error, $s = 1.95$ cm




(d) Std, $s = 1.95$ cm

| $s$ | $R(s, \bar{\tau})$ | $\mathcal{E}(s, \bar{\tau})$ |
|---|---|---|
| 0.08 | 16 | 1.00e-04 |
| 0.23 | 15 | 2.00e-04 |
| 0.39 | 14 | 4.00e-04 |
| 0.55 | 13 | 6.00e-04 |
| 0.70 | 13 | 8.00e-04 |
| 0.86 | 12 | 1.10e-03 |
| 1.01 | 11 | 1.50e-03 |
| 1.17 | 10 | 2.10e-03 |
| 1.33 | 9 | 3.10e-03 |
| 1.48 | 8 | 4.50e-03 |
| 1.64 | 8 | 6.50e-03 |
| 1.79 | 7 | 8.20e-03 |
| 1.95 | 7 | 1.07e-02 |
| 2.11 | 6 | 1.23e-02 |
| 2.26 | 6 | 1.39e-02 |

*Constantine, Gleich,*
*Hou & Templeton arXiv 2013.*

# Dynamic Mode Decomposition

Dynamic mode decomposition of a rectangular supersonic screeching jet

Joseph W. Nichols

July 20, 2012

A

# Is this BIG Data?

BIG Data has two properties

- too big for one hard drive
- ‘skewed’ distribution

BIG Data = “Big Internet Giant” Data

BIG Data = “Big In’Gineering” Data

“Engineering”

A matrix $\boldsymbol{A} : m \times n, m \geq n$
is tall and skinny when $O(n^2)$
work and storage is “cheap”
compared to $m$.

-- Austin Benson

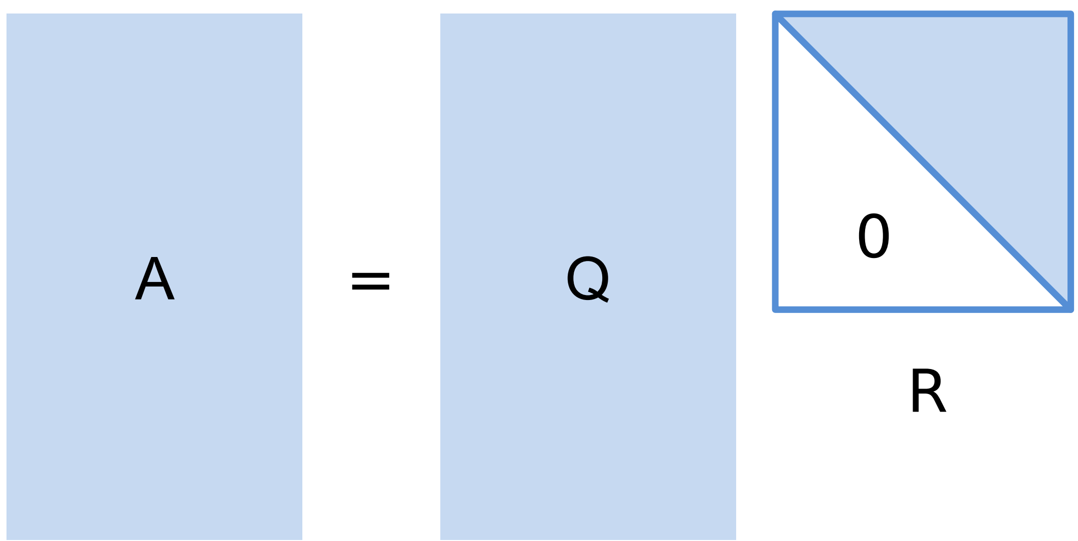
# Quick review of QR

Let $\boldsymbol{A} : m \times n$, $m \geq n$, real

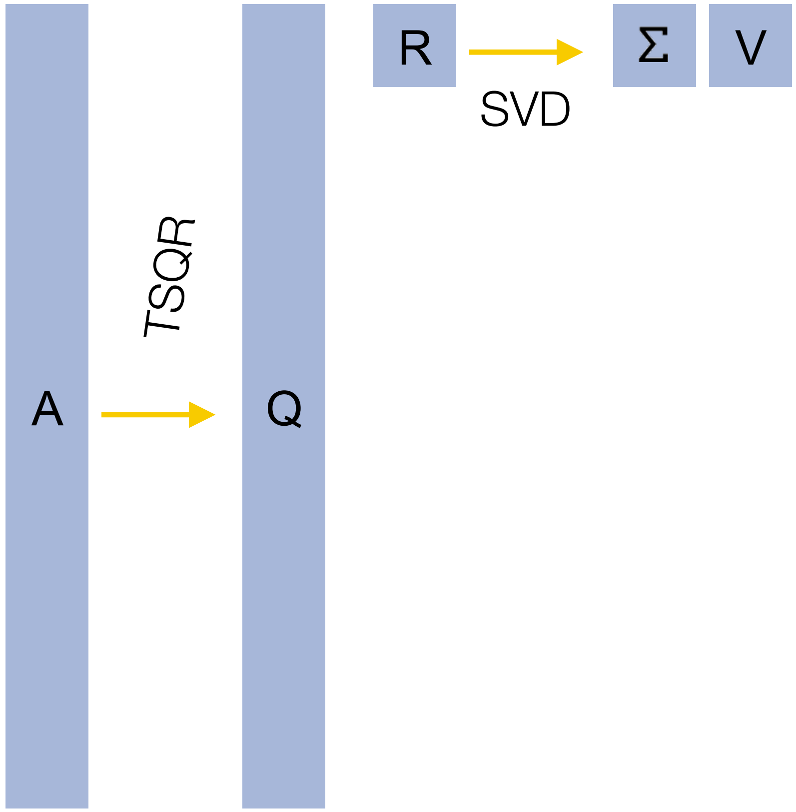
$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$

$\boldsymbol{Q}$ is $m \times n$ orthogonal ($\boldsymbol{Q}^T \boldsymbol{Q} = \boldsymbol{I}$)

$\boldsymbol{R}$ is $n \times n$ upper triangular



# Tall-and-skinny SVD and RSVD




Let $\boldsymbol{A}$ : $m \times n$, $m \geq n$, real

$$\boldsymbol{A} = \boldsymbol{U\Sigma V^T}$$

$\boldsymbol{U}$ is $m \times n$ orthogonal

$\boldsymbol{\Sigma}$ is $m \times n$ nonneg, diag.

$\boldsymbol{V}$ is $n \times n$ orthogonal

There are good MPI implementations.

What's left to do?

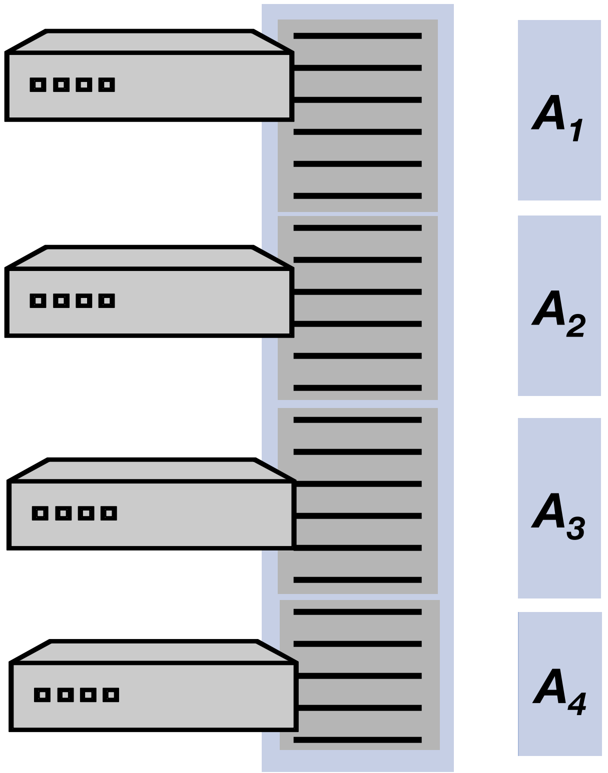
Moving data to an MPI cluster may be infeasible or costly

# How to store tall-and-skinny matrices in Hadoop

$A : m \times n$, $m \gg n$

Key is an arbitrary row-id
Value is the $1 \times n$ array
for a row (or $b \times n$ block)

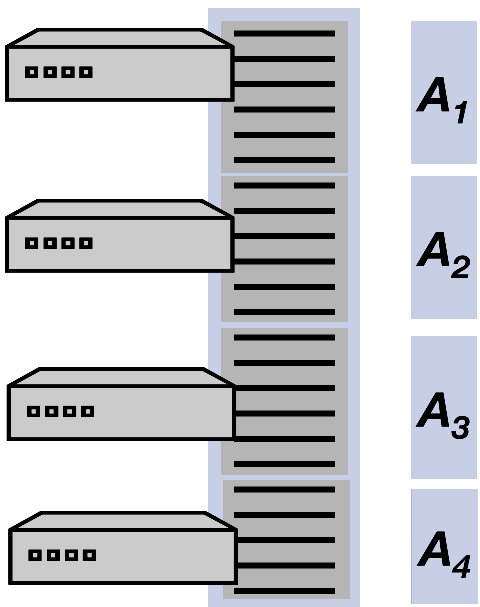
Each submatrix $A_i$ is an
the input to a map task.



# Still, isn't this easy to do?

**Current MapReduce algs** use the normal equations

$$\mathbf{A} = \mathbf{QR} \qquad \mathbf{A}^T\mathbf{A} \xrightarrow{\text{Cholesky}} \mathbf{R}^T\mathbf{R} \qquad \mathbf{Q} = \mathbf{A}\mathbf{R}^{-1}$$




**Map**
$A_{ii}$ to $A_i^T A_i$

**Reduce**
$R^T R = \text{Sum}(A_i^T A_i)$

**Map 2**
$A_{ii}$ to $A_i R^{-1}$

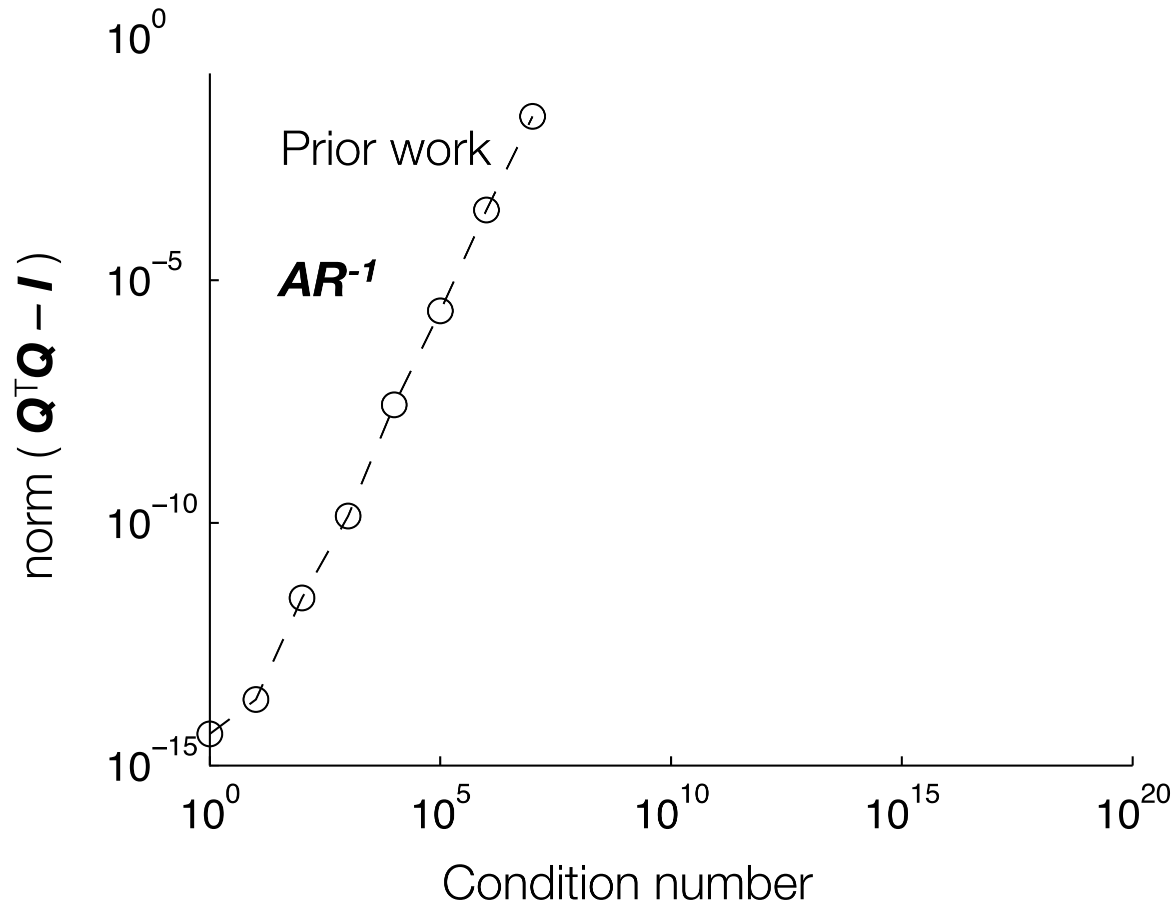
**Two problems**

R inaccurate if A ill-conditioned

Q not numerically orthogonal (Householder assures this)

# Numerical stability was a problem for prior approaches

Previous methods couldn't ensure that the matrix Q was orthogonal



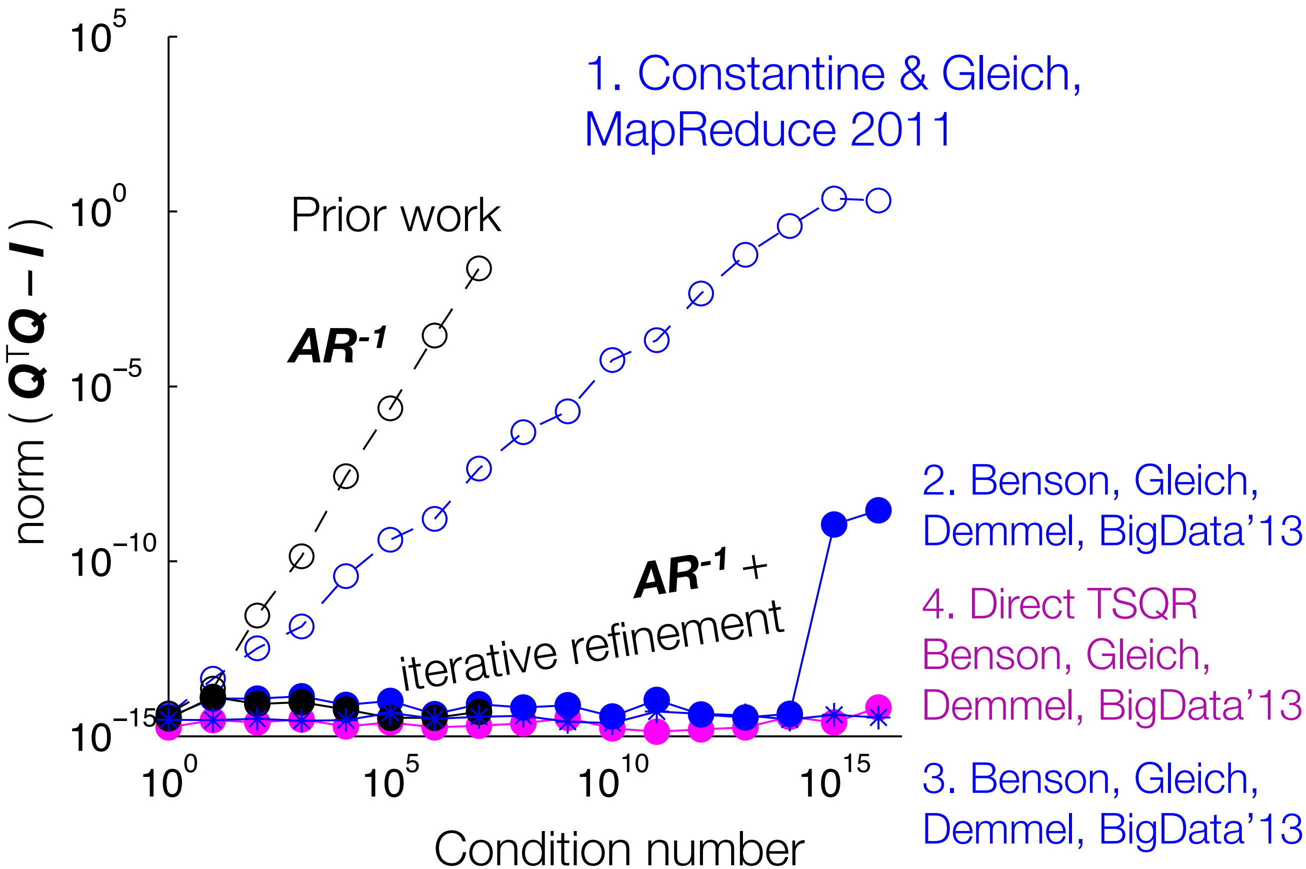
# Four things that are better

1. A simple algorithm to compute R accurately. (but doesn't help get Q orthogonal).
2. "Fast algorithm" to get Q numerically orthogonal in most cases.
3. Multi-pass algorithm to get Q numerically orthogonal in virtually all cases.
4. A direct algorithm for a numerically orthogonal Q in all cases.

Constantine & Gleich MapReduce 2011
Benson, Gleich & Demmel IEEE BigData 2013

# Numerical stability was a problem for prior approaches

Previous methods couldn't ensure that the matrix Q was orthogonal



# MapReduce is great for TSQR!
# You don't need $A^T A$

**Data** A tall and skinny (TS) matrix by rows

Input 500,000,000-by-50 matrix
Each record 1-by-50 row
HDFS Size 183.6 GB

Time to compute read A **253 sec.** write A **848 sec.**
Time to compute R in qr(A) **526 sec.** w/ **$Q=AR^{-1}$ 1618 sec**.
Time to compute Q in qr(A) **3090 sec.** (numerically stable)

```
git clone https://github.com/arbenson/mrtsqr
```

# Communication avoiding QR (Demmel et al. 2008)

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix}$$

$$A = \underbrace{\begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & Q_3 & \\ & & & Q_4 \end{bmatrix}}_{8n \times 4n} \underbrace{\begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix}}_{4n \times n}$$

First, do QR factorizations of each local matrix $\boldsymbol{A}_i$

Second, compute a QR factorization of the new "R"

$$A = \overbrace{\underbrace{\begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & Q_3 & \\ & & & Q_4 \end{bmatrix}}_{8n \times 4n} \underbrace{\tilde{Q}}_{4n \times n}}^{=Q} \underbrace{\tilde{R}}_{n \times n}$$

# Serial QR factorizations (Demmel et al. 2008)

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix}$$

Compute QR of $\boldsymbol{A}_1$,
read $\boldsymbol{A}_2$, update QR, …

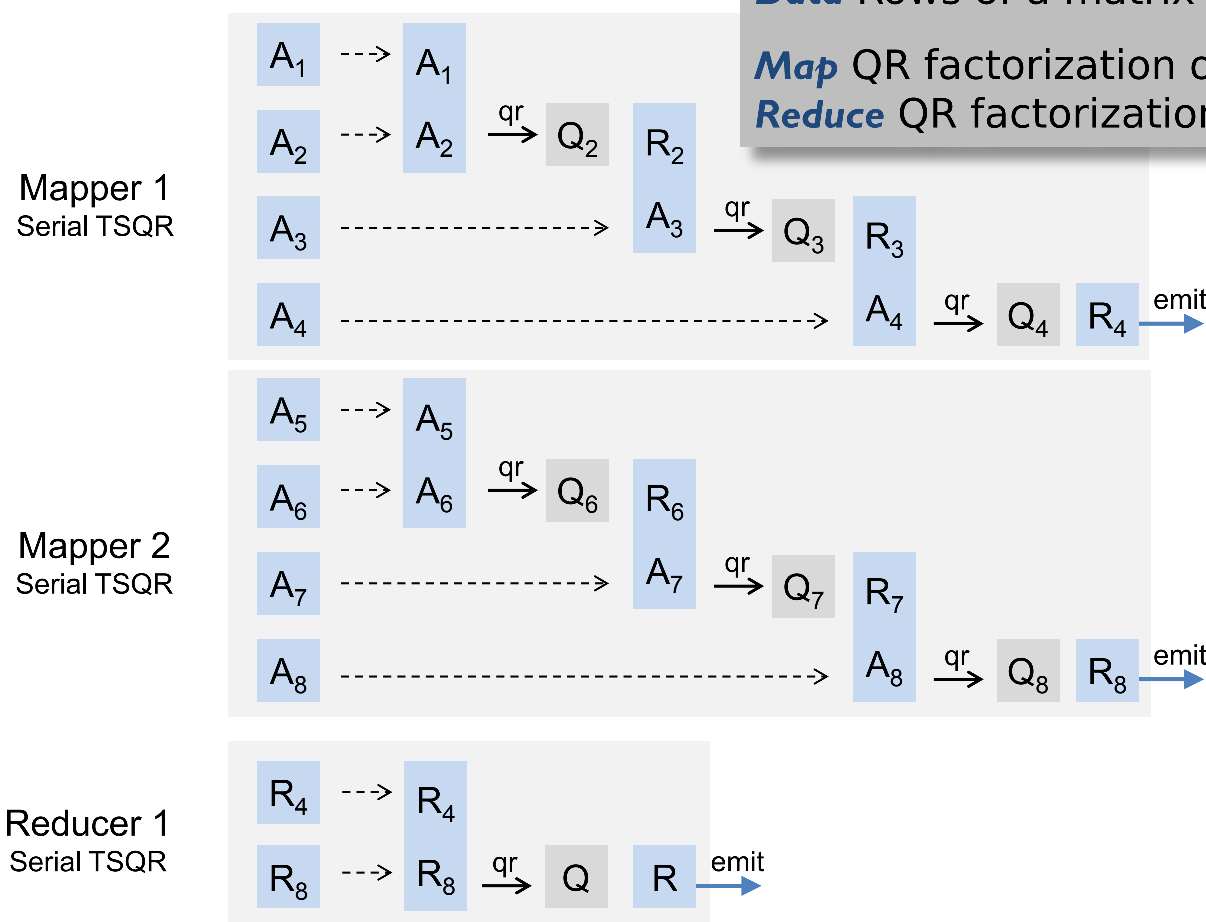
$$A_1 = Q_1 R_1; \begin{bmatrix} R_1 \\ A_2 \end{bmatrix} = Q_2 R_2; \begin{bmatrix} R_2 \\ A_3 \end{bmatrix} = Q_3 R_3; \begin{bmatrix} R_3 \\ A_4 \end{bmatrix} = Q_4 R_4$$

$$A = \overbrace{\underbrace{\begin{bmatrix} Q_1 & & & \\ & I_{2n} & & \\ & & I_{2n} & \\ & & & I_{2n} \end{bmatrix}}_{8n \times 7n} \underbrace{\begin{bmatrix} Q_2 & & \\ & I_{2n} & \\ & & I_{2n} \end{bmatrix}}_{7n \times 5n} \underbrace{\begin{bmatrix} Q_3 & \\ & I_{2n} \end{bmatrix}}_{5n \times 3n} \underbrace{Q_4}_{3n \times n}}^{=Q} \underbrace{R}_{n \times n}$$

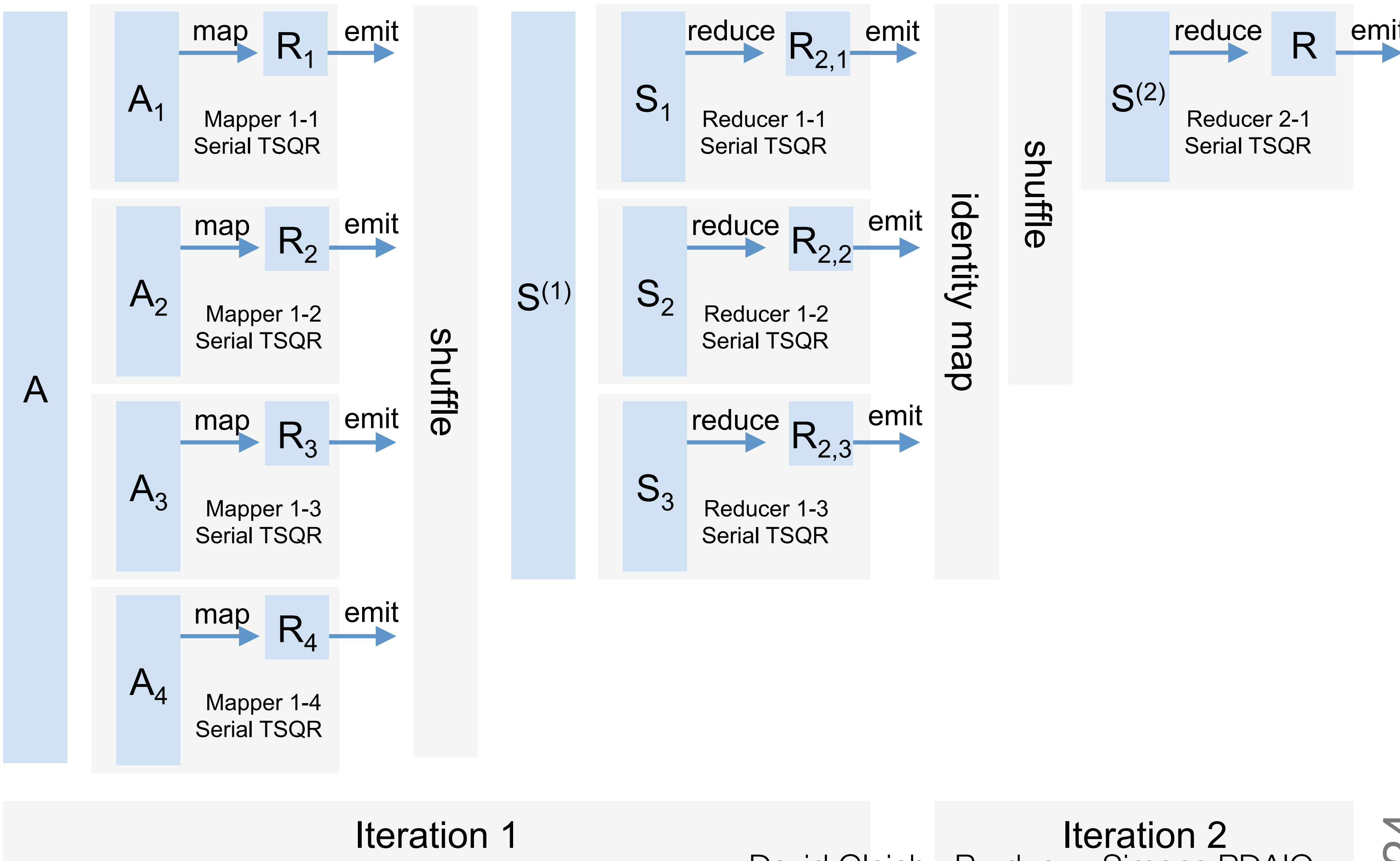
# Communication avoiding QR (Demmel et al. 2008) on MapReduce (Constantine and Gleich, 2011)

**Algorithm**
***Data*** Rows of a matrix

***Map*** QR factorization of rows
***Reduce*** QR factorization of rows



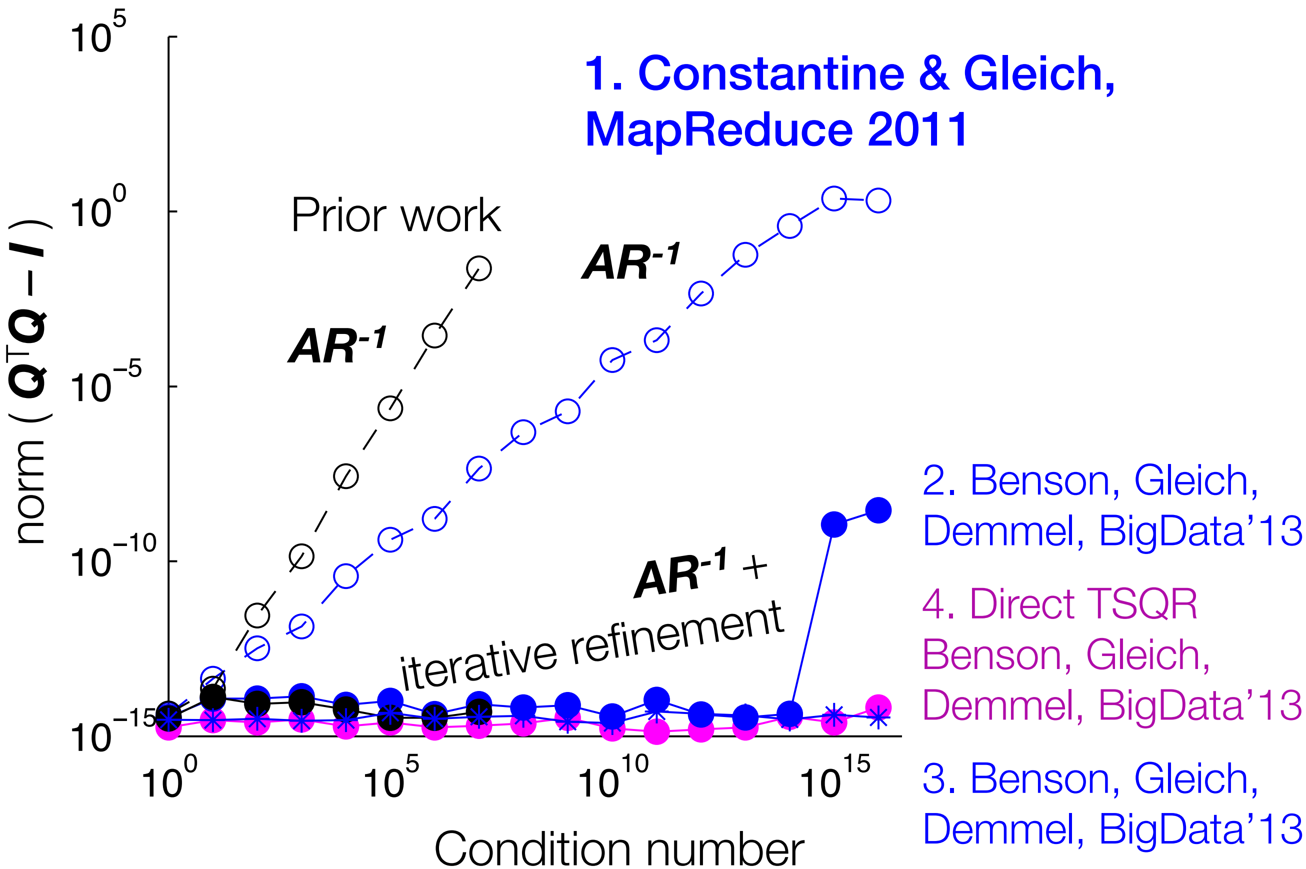
# Too many maps cause too much data to one reducer!



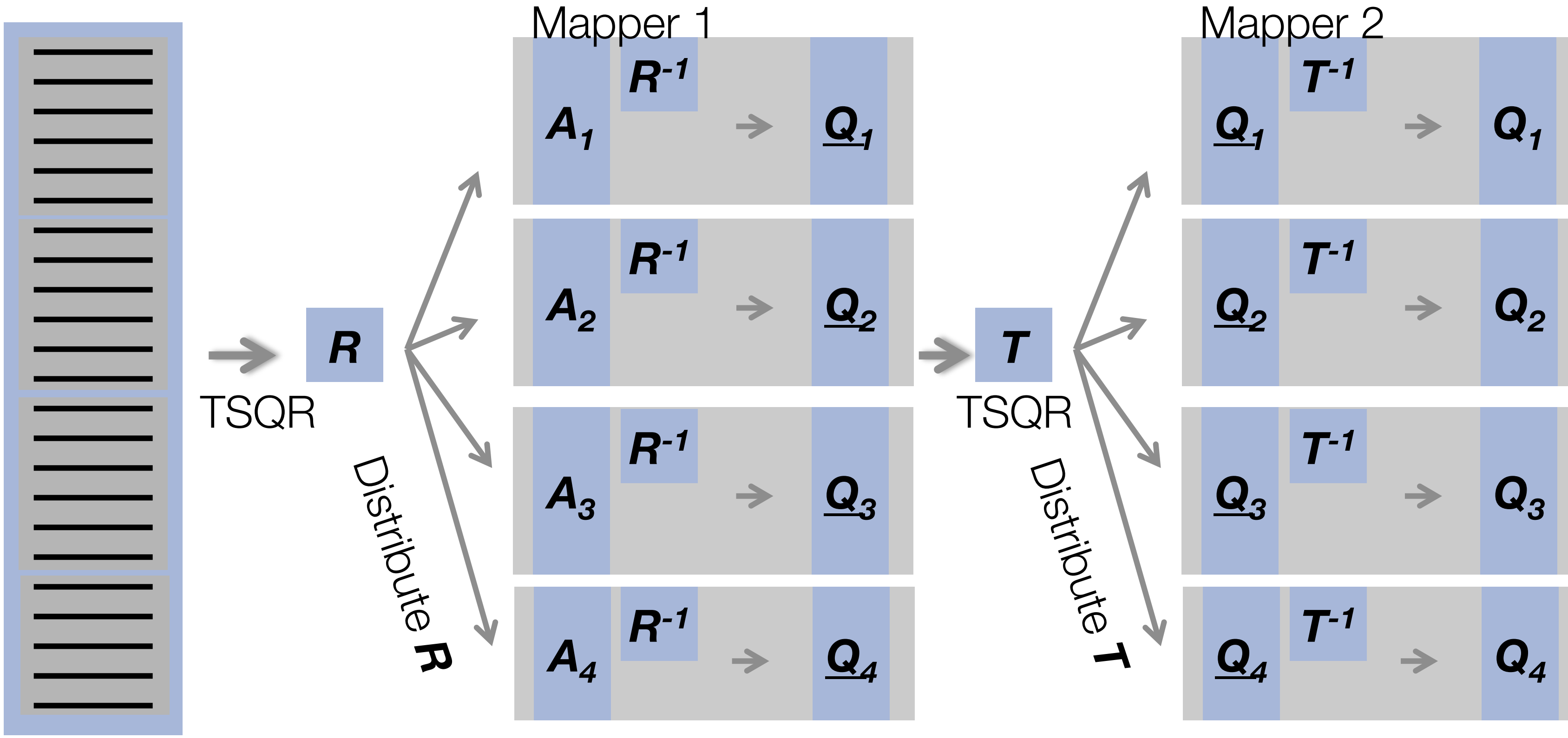
# Getting Q

# Numerical stability was a problem for prior approaches

Previous methods couldn't ensure that the matrix Q was orthogonal

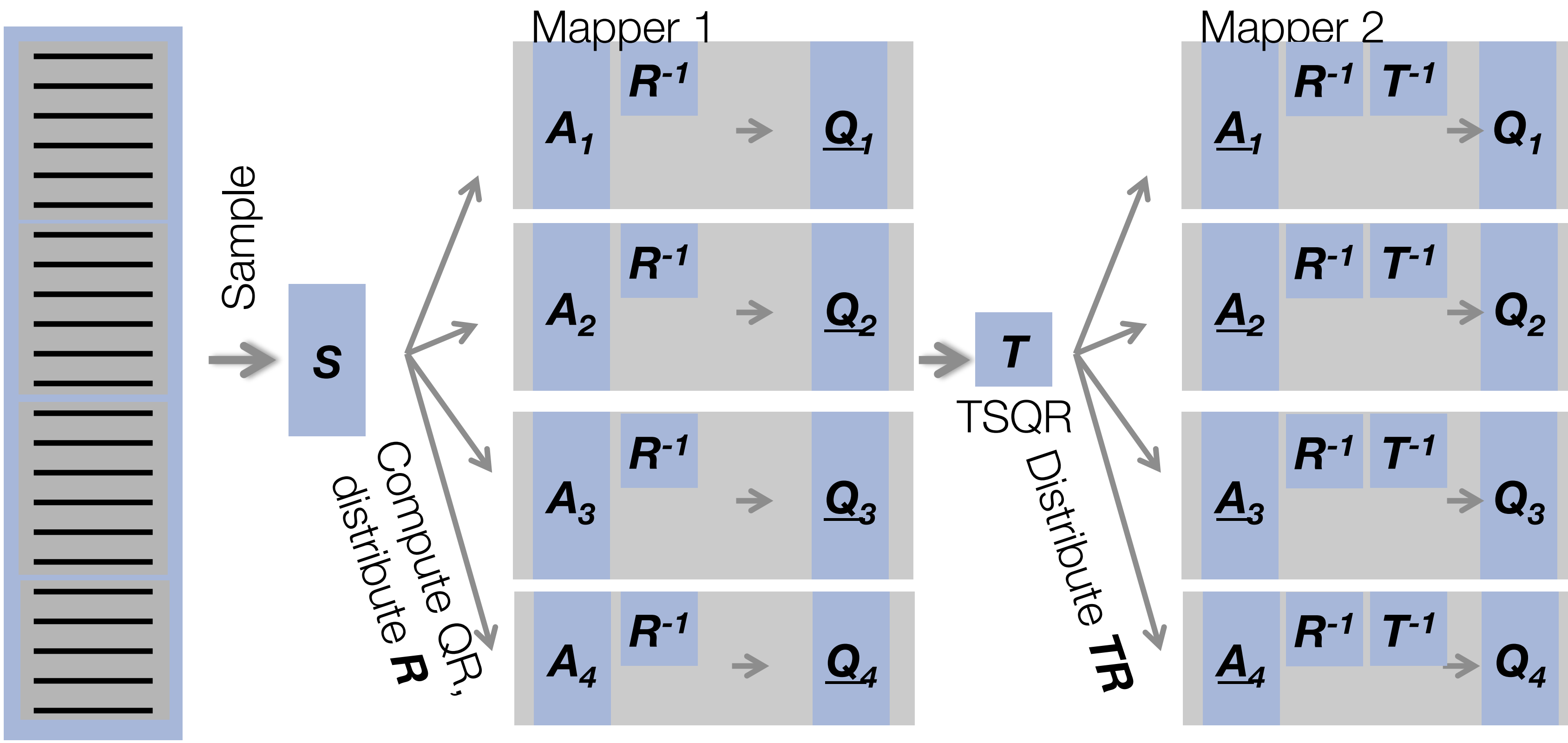


# Iterative refinement helps




Iterative refinement is like using Newton's method to solve Ax = b. It's folklore that "two iterations of iterative refinement are enough"

# What if iterative refinement is too slow?

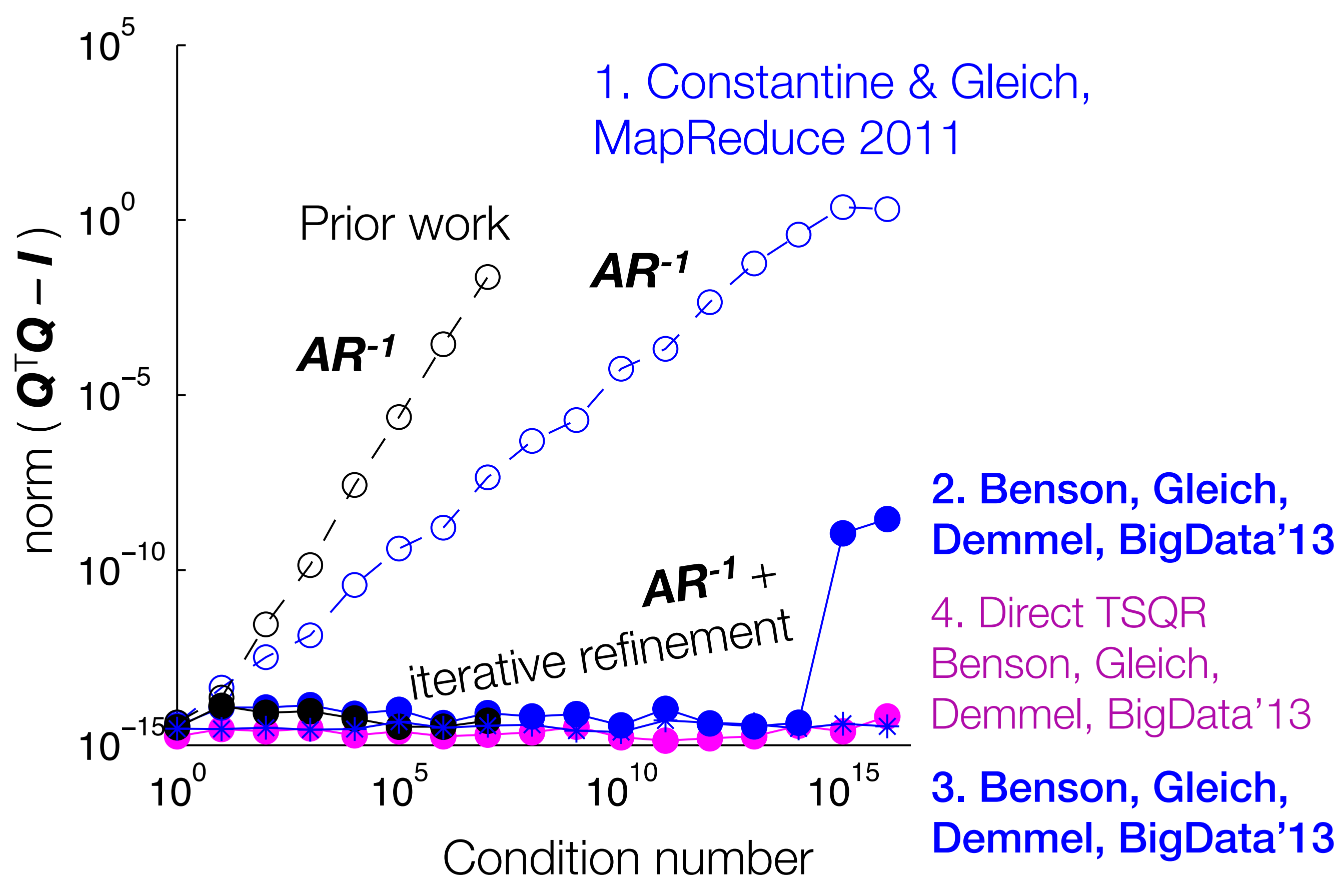



Estimate the "norm" by ***S***

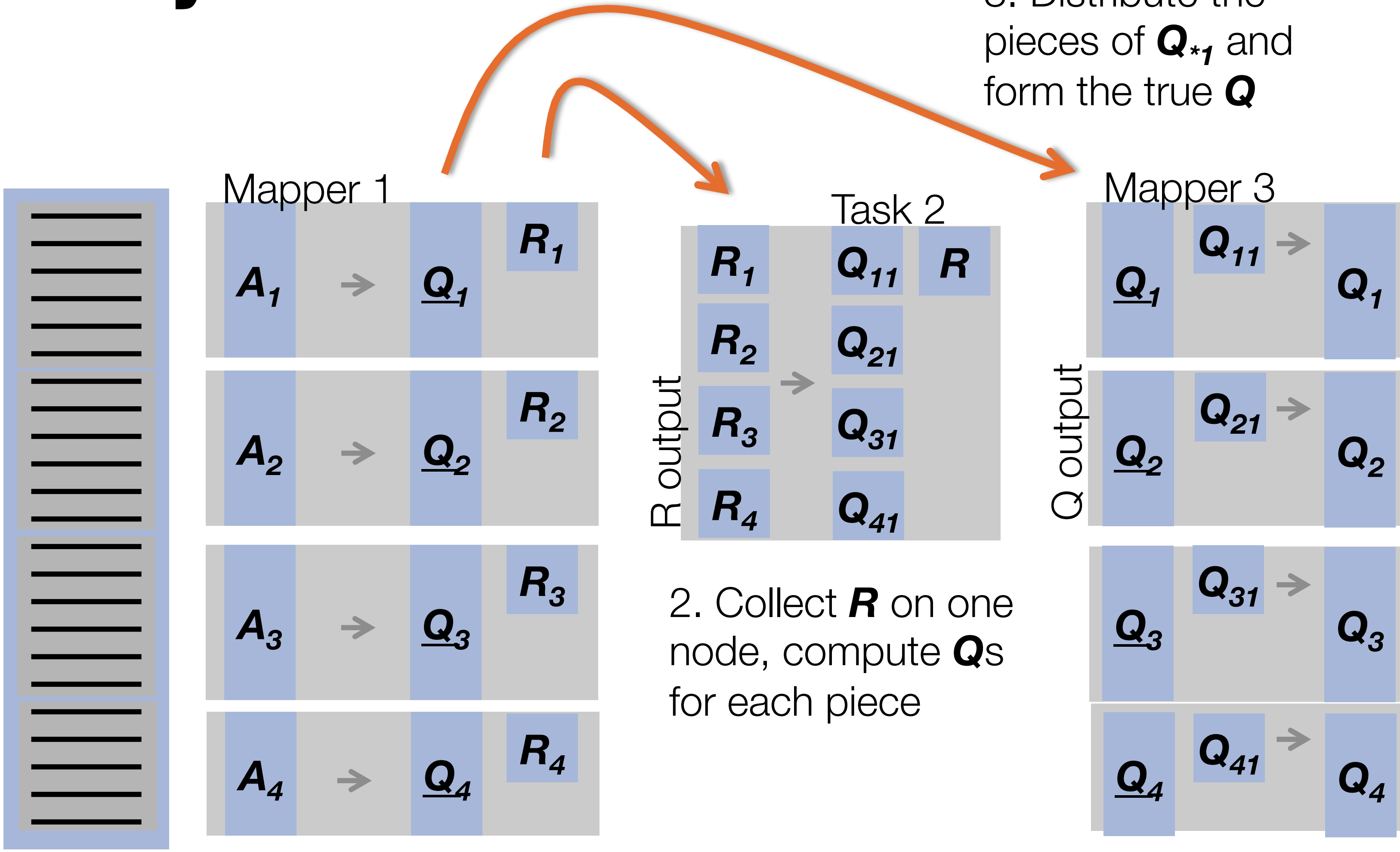
Based on recent work by "random matrix" community on approximating QR with a random subset of rows. Also assumes that you can get a subset of rows "cheaply" – possible, but nontrivial in Hadoop.

# Numerical stability was a problem for prior approaches

Previous methods couldn't ensure that the matrix Q was orthogonal



# Recreate Q by storing the history of the factorization




1. Output local $\underline{Q}$ and $R$ in separate files

2. Collect $R$ on one node, compute $Q$s for each piece

3. Distribute the pieces of $Q_{*1}$ and form the true $Q$

# Theoretical lower bound on runtime for a few cases on our small cluster

Model

| Rows | Cols | Old | R-only + no IR | R-only + PIR | R-only + IR | Direct TSQR |
|---|---|---|---|---|---|---|
| 4.0B | 4 | 1803 | 1821 | 1821 | 2343 | 2525 |
| 2.5B | 10 | 1645 | 1655 | 1655 | 2062 | 2464 |
| 0.6B | 25 | 804 | 812 | 812 | 1000 | 1237 |
| 0.5B | 50 | 1240 | 1250 | 1250 | 1517 | 2103 |

Actual

| Rows | Cols | Old | R-only + no IR | R-only + PIR | R-only + IR | Direct TSQR |
|---|---|---|---|---|---|---|
| 4.0B | 4 | 2931 | 3460 | 3620 | 4741 | 6128 |
| 2.5B | 10 | 2508 | 2509 | 3354 | 4034 | 4035 |
| 0.6B | 25 | 1098 | 1104 | 1476 | 2006 | 1910 |
| 0.5B | 50 | 921 | 1618 | 1960 | 2655 | 3090 |

All values in seconds

Only two params needed – read and write bandwidth for the cluster – in order to derive a performance model of the algorithm. This simple model is almost within a factor of two of the true runtime. (10-node cluster, 60 disks)

# Papers

Constantine & Gleich, MapReduce 2011

Benson, Gleich & Demmel, BigData'13

Constantine & Gleich, ICASSP 2012

Constantine, Gleich, Hou & Templeton, arXiv 2013

# Code

https://github.com/arbenson/mrtsqr

https://github.com/dgleich/simform

## Questions?

# BIG

Bloody Imposing Graphs

Building Impressions of Groundtruth

Blockwise Independent Guesses

Best Implemented at Google