Tall-and-skinny QRs and SVDs in MapReduce

David F. Gleich
Computer Science
Purdue University

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

Proportion of temp. > 475 K

Bubble radius

Insulator regime

David Gleich · Purdue Simons PDAIO
Each simulation is a column

5B-by-64 matrix
2.2TB

Extract 128 x 128 face to laptop

S VΤ

A → U → UF S VΤ
Non-insulator regime

Proportion of temp. > 475 K

Bubble radius

Insulator regime

David Gleich · Purdue  Simons PDAIO
ROM and apply to the far face for the UQ study in Section 4.4. A reduction in the data enabled us to use a laptop to compute the coefficients of the TSSVD, and compute predictions and errors. The preprocessing steps took approximately 20 cpu-hours consumed via these calculations—compared to the 131,072 hours it would have taken to compute the full solution using TSSVD. Also, during our computations, we observed failures in hard disk drives and issues with memory usage. This caused entire nodes to fail. Given that the cluster has 40 cores, there was at most 33,000 cores available to run 4TB of data. Jobs with sizes ranging between 100GB and 2TB of data sometimes ran concurrently. Generating 8192 meshes with different material properties and running independent simulations of approximating the full solution with a reduced-order model? Why not just use a response surface to interpolate the means of each of the two quantities of interest over a range of bubble radii. We use the quantities of interest at the final time for bubble radii, bubble boundaries and larger near the face containing the heat source. Visualizing the varying conductivity fields took approximately twenty minutes to construct using Cubit after substantial optimizations.

### Table 4.1: The split and the correspondence

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

(c) Error, $s = 1.95$ cm
(d) Std, $s = 1.95$ cm

---

Constantine, Gleich, Hou & Templeton arXiv 2013.

David Gleich · Purdue · Simons PDAIO
Dynamic mode decomposition of a rectangular supersonic screeching jet

Joseph W. Nichols

July 20, 2012
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 $\mathbf{A} : m \times n, m \geq n$ is tall and skinny when $O(n^2)$ work and storage is “cheap” compared to $m$.

-- Austin Benson
Quick review of QR

Let $A : m \times n, m \geq n$, real

$A = QR$

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

$R$ is $n \times n$ upper triangular
Tall-and-skinny SVD and RSVD

Let $A : m \times n, m \geq n$, real

$$A = U \Sigma V^T$$

$U$ is $m \times n$ orthogonal

$\Sigma$ is $m \times n$ nonneg, diag.

$V$ 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

\[ A = QR \quad A^T A \xrightarrow{\text{Cholesky}} R^T R \quad Q = AR^{-1} \]

Map

\[ A_{ii} \text{ to } A_i^TA_i \]

Reduce

\[ R^TR = \text{Sum}(A_i^TA_i) \]

Map 2

\[ A_{ii} \text{ to } A_iR^{-1} \]

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.

\[ \text{norm} (Q^T Q - I) \]

\[ AR^{-1} \]

Prior work

David Gleich · Purdue  Simons PDAIO
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.
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 $\text{qr}(A)$ 526 sec. w/ $Q=AR^{-1}$ 1618 sec.
Time to compute $Q$ in $\text{qr}(A)$ 3090 sec. (numerically stable)

**git clone** https://github.com/arbenson/mrtsqr
Communication avoiding QR (Demmel et al. 2008)

First, do QR factorizations of each local matrix $A_i$

Second, compute a QR factorization of the new "R"
Serial QR factorizations (Demmel et al. 2008)

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

Compute QR of \( A_1 \), read \( A_2 \), update QR, ...

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

---

**Mapper 1**
Serial TSQR

- $A_1 \rightarrow A_1
- A_2 \rightarrow A_2 \rightarrow Q_2 \rightarrow R_2
- A_3 \rightarrow A_3 \rightarrow Q_3 \rightarrow R_3
- A_4 \rightarrow A_4 \rightarrow Q_4 \rightarrow R_4$ emit

**Mapper 2**
Serial TSQR

- $A_5 \rightarrow A_5
- A_6 \rightarrow A_6 \rightarrow Q_6 \rightarrow R_6
- A_7 \rightarrow A_7 \rightarrow Q_7 \rightarrow R_7
- A_8 \rightarrow A_8 \rightarrow Q_8 \rightarrow R_8$ emit

**Reducer 1**
Serial TSQR

- $R_4 \rightarrow R_4
- R_8 \rightarrow R_8 \rightarrow Q \rightarrow R$ emit
Too many maps cause too much data to one reducer!

Iteration 1

A

mapper

A1

Mapper 1-1
Serial TSQR

map

R1

emit

S1

Reducer 1-1
Serial TSQR

reduce

R2,1

emit

A2

Reducer 1-2
Serial TSQR

mapper

A2

map

R2

emit

S1

Reducer 1-2
Serial TSQR

reduce

R2,2

emit

A3

Reducer 1-3
Serial TSQR

mapper

A3

map

R3

emit

S2

Reducer 1-3
Serial TSQR

reduce

R2,3

emit

A4

Reducer 1-4
Serial TSQR

mapper

A4

map

R4

emit

S3

Reducer 2-1
Serial TSQR

reduce

S(2)

emit

Iteration 2

Simons PDAIO

David Gleich · Purdue
Getting Q
Numerical stability was a problem for prior approaches

Previous methods couldn’t ensure that the matrix $Q$ was orthogonal.

1. Constantine & Gleich, MapReduce 2011

2. Benson, Gleich, Demmel, BigData’13

3. Benson, Gleich, Demmel, BigData’13

4. Direct TSQR

Benson, Gleich, Demmel, BigData’13
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?

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.

1. Constantine & Gleich, MapReduce 2011

2. Benson, Gleich, Demmel, BigData’13

3. Benson, Gleich, Demmel, BigData’13

4. Direct TSQR
Benson, Gleich, Demmel, BigData’13

David Gleich · Purdue Simons PDAIO
Recreate Q by storing the history of the factorization

Mapper 1

A_1 \rightarrow Q_1 \rightarrow R_1

A_2 \rightarrow Q_2 \rightarrow R_2

A_3 \rightarrow Q_3 \rightarrow R_3

A_4 \rightarrow Q_4 \rightarrow R_4

Mapper 3

Q_1 \rightarrow Q_{11} \rightarrow Q_1

Q_2 \rightarrow Q_{21} \rightarrow Q_2

Q_3 \rightarrow Q_{31} \rightarrow Q_3

Q_4 \rightarrow Q_{41} \rightarrow Q_4

Task 2

R output

R_1 \rightarrow Q_{11} \rightarrow R

R_2 \rightarrow Q_{21} \rightarrow Q_3

R_3 \rightarrow Q_{31} \rightarrow Q_4

R_4 \rightarrow Q_{41} \rightarrow

Q output

Q_1 \rightarrow Q_{11} \rightarrow Q_1

Q_2 \rightarrow Q_{21} \rightarrow Q_2

Q_3 \rightarrow Q_{31} \rightarrow Q_3

Q_4 \rightarrow Q_{41} \rightarrow Q_4

1. Output local 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

<table>
<thead>
<tr>
<th>Rows</th>
<th>Cols</th>
<th>Old</th>
<th>R-only + no IR</th>
<th>R-only + PIR</th>
<th>R-only + IR</th>
<th>Direct TSQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0B</td>
<td>4</td>
<td>1803</td>
<td>1821</td>
<td>1821</td>
<td>2343</td>
<td>2525</td>
</tr>
<tr>
<td>2.5B</td>
<td>10</td>
<td>1645</td>
<td>1655</td>
<td>1655</td>
<td>2062</td>
<td>2464</td>
</tr>
<tr>
<td>0.6B</td>
<td>25</td>
<td>804</td>
<td>812</td>
<td>812</td>
<td>1000</td>
<td>1237</td>
</tr>
<tr>
<td>0.5B</td>
<td>50</td>
<td>1240</td>
<td>1250</td>
<td>1250</td>
<td>1517</td>
<td>2103</td>
</tr>
</tbody>
</table>

Model

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

Actual

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

Questions?

BIG

Bloody Imposing Graphs
Building Impressions of Groundtruth
Blockwise Independent Guesses

Best Implemented at Google

Questions?

BIG

Bloody Imposing Graphs
Building Impressions of Groundtruth
Blockwise Independent Guesses

Best Implemented at Google

Code

https://github.com/arbenson/mrtsqr
https://github.com/dgleich/simform

David Gleich · Purdue · Simons PDAIO