QR Factorization in Parallel

Jacque Bush   Paul Torres

December 19, 2012

What is a QR Decomposition?

Any matrix $A \in \mathbb{C}^{m\times n}$ has a QR factorization, $Q \in \mathbb{C}^{m\times m}$ a unitary orthogonal matrix and $R \in \mathbb{C}^{m\times n}$ an upper triangular matrix. Since Q is unitary

$$\det(Q) = \pm 1 \quad \text{and} \quad Q^* Q = I.$$

If $m \geq n$ then R has the following form,

$$R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}.$$

The factorization

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

$\hat{Q} \in \mathbb{C}^{m\times n}$ and $\hat{R} \in \mathbb{C}^{n\times n}$ is called the reduced QR factorization.

Why QR?

QR decompositions can be used for many things:

- Fundamental Part of QR algorithm

Algorithm 1.1 The QR Algorithm (without shifts)

$A^{(0)} = A$
for $k = 1, 2, \cdots$ **do**
  $Q^{(k)}R^{(k)} = A^{(k-1)}$
  $A^{(k)} = R^{(k)}Q^{(k)}$
end for

- Least Squares Problem
- Other Matrix Factorizations
- Finding Eigenvalues and Eigenvectors of A

Navie QR Factorization

Three Different Ways to perform QR Factorization:

1. Gram-Schmidt
   - Fun Fact: This method is used as a proof that QR factorizations exists.
   - Can be unstable for matrices with almost linearly dependent columns.

Navie QR Factorization

Three Different Ways to perform QR Factorization:

1. Gram-Schmidt
    - Fun Fact: This method is used as a proof that QR factorizations exists.
    - Can be unstable for matrices with almost linearly dependent columns.
2. Givens Rotations
    - Zeros out one element at a time.
    - Can be slow if Matrix is not sparse.

Navie QR Factorization

Three Different Ways to perform QR Factorization:

1. Gram-Schmidt
    - Fun Fact: This method is used as a proof that QR factorizations exists.
    - Can be unstable for matrices with almost linearly dependent columns.
2. Givens Rotations
    - Zeros out one element at a time.
    - Can be slow if Matrix is not sparse.
3. Householder Reflections
    - Zeros out a whole column a time.
    - We used this algorithm as a base for our code.

Householder Reflections

Householder Reflections are special unitary matrices P_i such that,

$$\underset{A}{\begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{pmatrix}} \underset{P_1}{\rightarrow} \underset{P_1 A}{\begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{pmatrix}} \underset{P_2}{\rightarrow} \underset{P_2 P_1 A}{\begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & x \end{pmatrix}} \underset{P_3}{\rightarrow} \underset{P_3 P_2 P_1 A}{\begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \end{pmatrix}}$$

where

$$P_i = I - 2\frac{v_i v_i^t}{v_i^t v_i}.$$

and

$$v_i(k) = \mathrm{sign}(a_{ii})||A_{k,i}||_2 e_1 + A_{k,i} \quad \text{if} \quad i \geq k \quad \text{else} \quad v_i(k) = 0$$

QR Factorization via Householder Reflections

Algorithm 2.1 Householder QR Factorization

for $k = 1$ to n **do**
  $x = A_{k:m,k}$
  $v_k = \text{sign}(x_1)||x||_2 e_1 + x$
  $v_k = v_k/||v_k||_2$
  $A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^* A_{k:m,k:n})$
end for

Notice that this algorithm does NOT produce both the Q and the R. To get the Q we would need to multiply all of the Householder Reflections together.

WY Representation of Q

The WY representation of Q writes the product of householder reflection matrices

$$Q_k = P_1 \cdots P_k$$

in the form

$$Q_k = I + W_k Y_k^T$$

where W_k and Y_k are n by k matrices and

$$P_i = I - 2\frac{v_i v_i^T}{v_i^T v_i}.$$

is a rank one update.

WY continued

Then

$$Q_k^T A = (I + Y_k W_k^T)A = A + Y_k W_k^T A$$

and

$$\begin{aligned}
Q_k = Q_{k-1}P_k &= (I + W_{k-1}Y_{k-1}^T)(I - \beta v_k v_k^T) \\
&= I + W_{k-1}Y_{k-1}^T - \beta Q_{k-1} v_k v_k^T \\
&= I + \begin{pmatrix} W_{k-1} & -\beta Q_{k-1} v_k \end{pmatrix} \begin{pmatrix} Y_{k-1}^T \\ v_k^T \end{pmatrix} \\
&= I + \begin{pmatrix} W_{k-1} & -\beta Q_{k-1} v_k \end{pmatrix} \begin{pmatrix} Y_{k-1} & v_k \end{pmatrix}^T \\
\Rightarrow \quad W_k &= \begin{pmatrix} W_{k-1} & -\beta Q_{k-1} v_k \end{pmatrix} \\
\text{and} \quad Y_k^T &= \begin{pmatrix} Y_{k-1}^T \\ v_k^T \end{pmatrix}.
\end{aligned}$$

Simple 3x3 WY example

Given a matrix A,

$$A = \begin{pmatrix} 3.83 & 9.15 & 3.86 \\ 8.88 & 7.93 & 4.92 \\ 7.77 & 3.35 & 6.49 \end{pmatrix}$$

Step 1: Compute v_1 where a_1 is the first column of A,

$$\begin{aligned} v_1 &= a_1 + \mathrm{sign}(a_{11}) e_1 ||a_1||_2 \\ &= \begin{pmatrix} 3.83 \\ 8.88 \\ 7.77 \end{pmatrix} + \begin{pmatrix} 12.39118 \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 16.22118 \\ 8.86 \\ 7.77 \end{pmatrix} \end{aligned}$$

Simple 3x3 WY example continued...

Step 2: Update v_1 and compute $w_1 = -2v_1$,

$$\begin{aligned} v_1 &= v_1/||v_1||_2 \\ &= \frac{1}{20.04991}\begin{pmatrix} 16.22118 \\ 8.86 \\ 7.77 \end{pmatrix} \\ &= \begin{pmatrix} 0.80903 \\ 0.44189 \\ 0.38753 \end{pmatrix} \end{aligned}$$

Insert into W and Y^t matrices,

$$W = \begin{pmatrix} -1.61807 & 0.0 & 0.0 \\ -0.88379 & 0.0 & 0.0 \\ -0.77506 & 0.0 & 0.0 \end{pmatrix}, Y^t = \begin{pmatrix} 0.80903 & 0.44189 & 0.38753 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix}$$

Simple 3x3 WY example continued...

Step 3: Compute Q_1 and Q_1^t:

$$\begin{aligned} Q_1^t &= (I + WY^t)^t \\ &= \begin{pmatrix} -0.30909 & -0.71502 & -0.62705 \\ -0.71502 & 0.60945 & -0.34249 \\ -0.62705 & -0.34249 & 0.69963 \end{pmatrix} \end{aligned}$$

Notice that

$$Q_1^t a_1 = \begin{pmatrix} -12.39118 \\ 0 \\ 0 \end{pmatrix}$$

Simple 3x3 WY example continued...

Step 4: Update a_2,

$$\begin{aligned} a_2 &= Q_1 a_2 \\ &= \begin{pmatrix} -10.59857 \\ -2.85687 \\ -6.10982 \end{pmatrix} \end{aligned}$$

Step 5: Compute v_2 where x is the new a_2 with with zeros above row 2,

$$\begin{aligned} v_2 &= x + \mathrm{sign}(x_2) e_2 ||x||_2 \\ &= \begin{pmatrix} 0 \\ -2.85687 \\ -6.10982 \end{pmatrix} - \begin{pmatrix} 0 \\ 6.744745 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 9.60161 \\ -6.10982 \end{pmatrix} \end{aligned}$$

Simple 3x3 WY example continued...

Step 6: Update v_2, and compute $w_2 = -2Q_1 v_2$,

$$\begin{aligned} v_2 &= 1/||v_2||_2 \\ &= \frac{1}{11.38072}\begin{pmatrix} 0 \\ 9.60161 \\ -6.10982 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ -0.84367 \\ -0.53685 \end{pmatrix} \end{aligned}$$

Insert into W and Y^t matrices,

$$W = \begin{pmatrix} -1.6180 & -1.8797 & 0.0 \\ -0.8837 & 0.6606 & 0.0 \\ -0.7750 & 0.1732 & 0.0 \end{pmatrix}, Y^t = \begin{pmatrix} 0.8090 & 0.4418 & 0.3875 \\ 0.0 & -0.8436 & -0.5368 \\ 0.0 & 0.0 & 0.0 \end{pmatrix}$$

Simple 3x3 WY example continued...

Step 7: Compute Q_2 and Q_2^t,

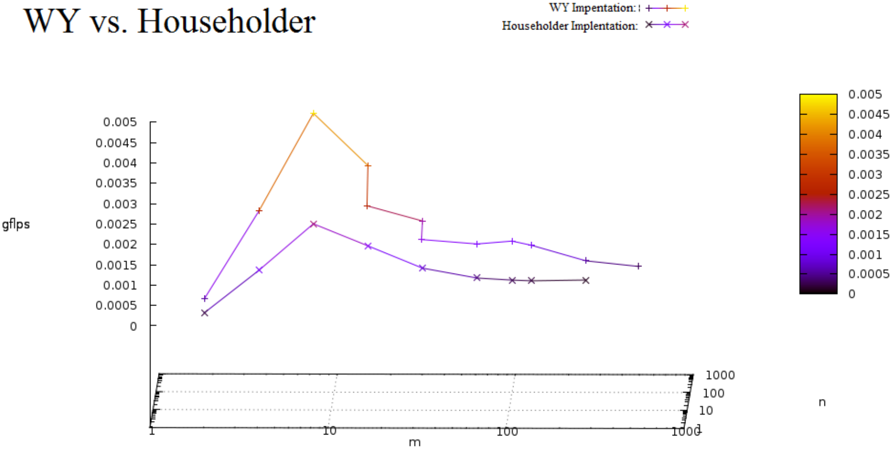
$$Q_2^t = (I + WY^t)^t$$
$$= \begin{pmatrix} -0.30909 & -0.715024 & -0.62705 \\ 0.87088 & 0.052117 & -0.488690 \\ 0.382119 & -0.697154 & 0.606604 \end{pmatrix}$$

Since A is square in this equation Q_2 is the final Q. In general the final Q_k would be when $k = m$, the height of A.
Step 8: Multiply A by Q^t to get final R,

$$R = Q^t A = \begin{pmatrix} -12.391182 & -10.59872 & -8.78062 \\ 0.0 & 6.74481 & 0.446449 \\ 0 & 0 & 1.9818806 \end{pmatrix}$$

Notice that $Q^t Q = I$, R is upper triangular and $QR = A$.

WY Preformance:



Problems with WY on its own:

1. Best Performance at size 8 by 8.
2. Can't handle large matrices.

Problems with WY on its own:

1. Best Performance at size 8 by 8.
2. Can't handle large matrices.
3. NO PARALLIZATION!!!!

Problems with WY on its own:

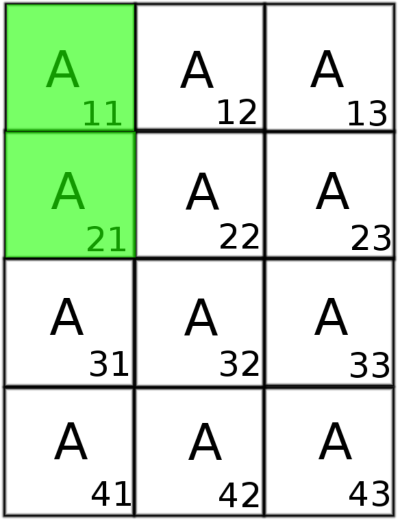
1. Best Performance at size 8 by 8.
2. Can't handle large matrices.
3. NO PARALLIZATION!!!!

SOLUTION: Block matrix A - This lead to our Blocked QR version 1.

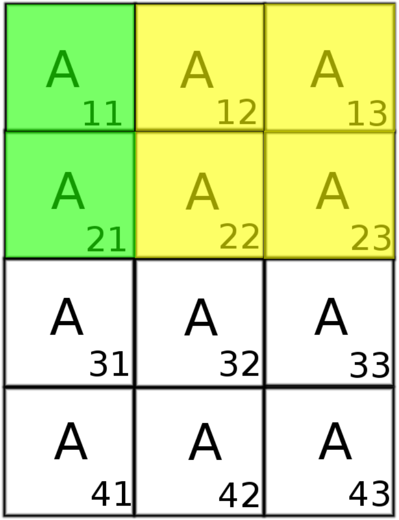
Given a matrix A:

$$\begin{array}{|c|c|c|}
\hline
A_{11} & A_{12} & A_{13} \\
\hline
A_{21} & A_{22} & A_{23} \\
\hline
A_{31} & A_{32} & A_{33} \\
\hline
A_{41} & A_{42} & A_{43} \\
\hline
\end{array}$$

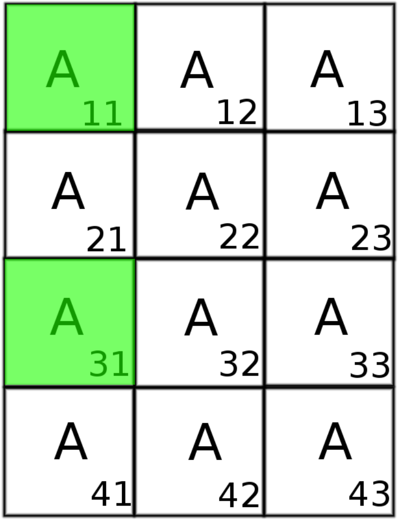
Step 1: Preform QR factorization on green blocks.



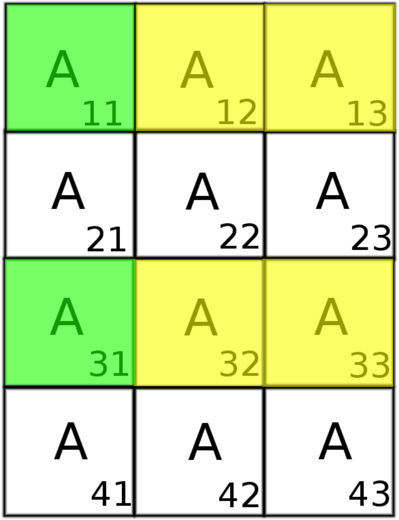
Step 2 and 3: Mutiply yellow blocks in parallel by Q^t obtained in pervious step and update Q matrix.



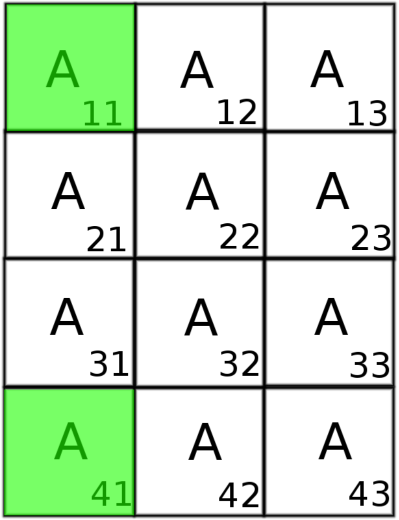
Step 1: Preform QR factorization on green blocks.



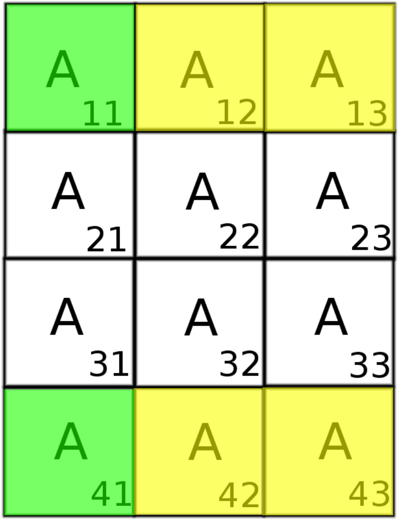
Step 2 and 3: Mutiply yellow blocks in parallel by Q^t obtained in pervious step and update Q matrix.



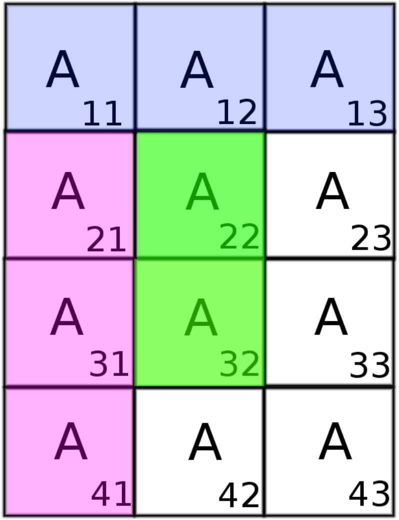
Step 1: Preform QR factorization on green blocks.



Step 2 and 3: Mutiply yellow blocks in parallel by Q^t obtained in pervious step and update Q matrix.



Step 1: Preform QR factorization on green blocks.



Problems with Version 1:

1. Not zeroing out blocks in parallel.
   If we have a thin and tall matrix this becomes problematic.

Problems with Version 1:

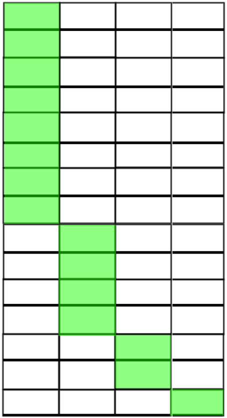
1. Not zeroing out blocks in parallel.
   If we have a thin and tall matrix this becomes problematic.
2. Not Utilizing L2 Cache sizes.

Problems with Version 1:

1. Not zeroing out blocks in parallel.
   If we have a thin and tall matrix this becomes problematic.
2. Not Utilizing L2 Cache sizes.

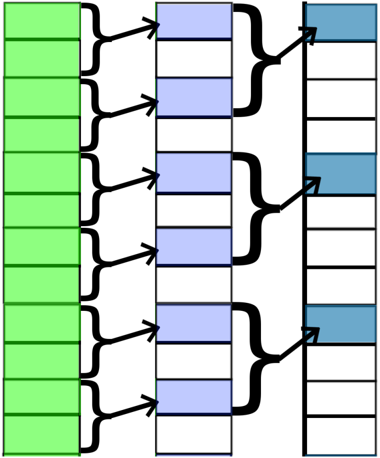
We created Tiled QR Factorization: version 2 to fix the first problem.

Step 1: Break up each column into sets of blocks below and including diagoanal.

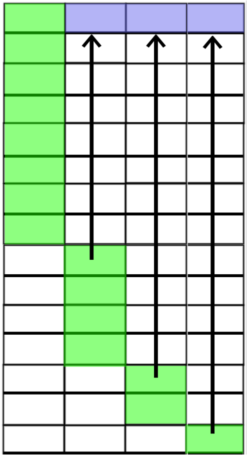


Note: This diagram denotes one column being broken up and zeroed out over multiple iterations.

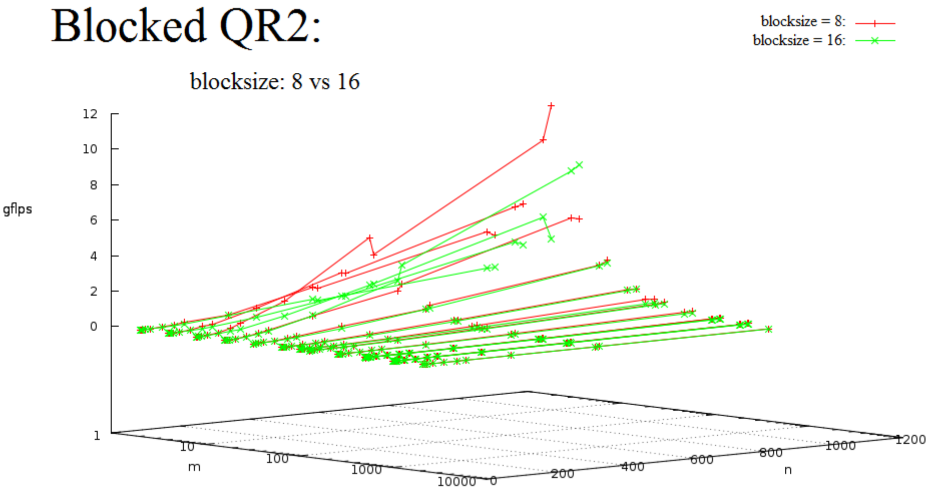
Step 2: Merge each set of blocks (green sets from pervious slide) using a binary tree.



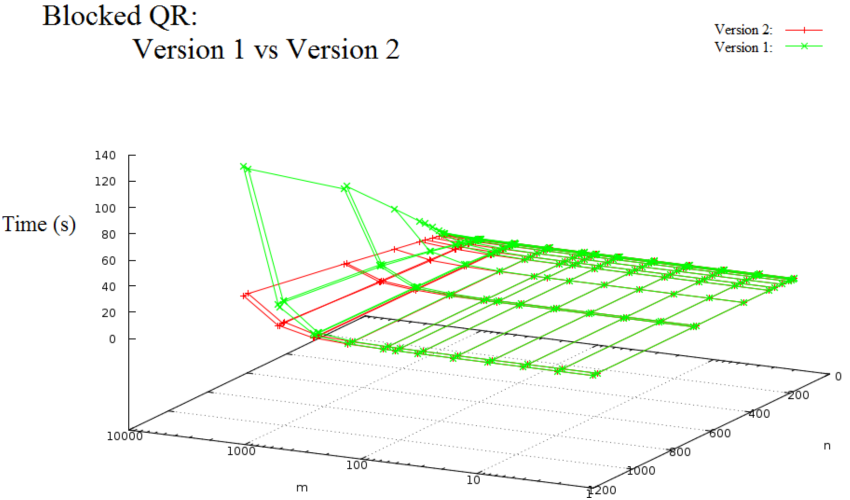
Step 3: Update root block at the after set has been merged.



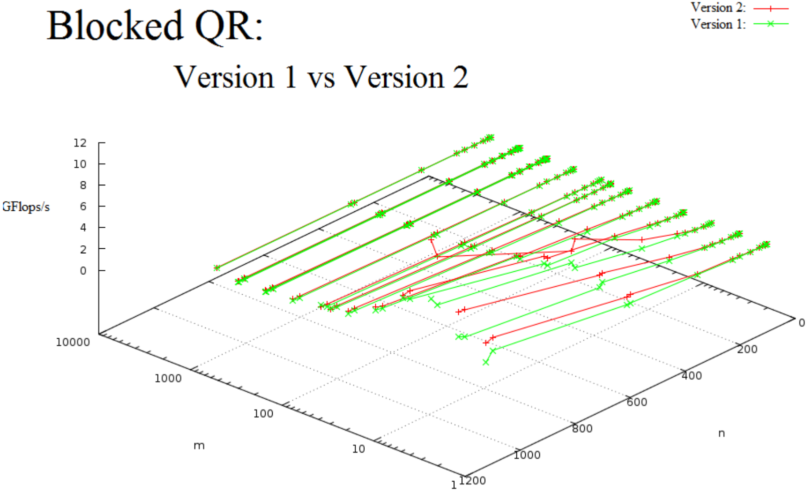
Study on block size:



Comparison Study: seconds on bowery



Comparison Study: GFlops per second on bowery



Futher Work:

- Utilize L2 Cahce sizes by adding another layer of blocking.

Futher Work:

- Utilize L2 Cahce sizes by adding another layer of blocking.
- Figure out a way to avoid bottlenecks that appear in Version 2.