Investigation into a Parallel Singular Value Decomposition

Travis Askham Steven Delong Michael Lewis

Courant Institute, New York

December 18, 2012

The Singular Value Decomposition

Given an $m \times n$ matrix A, its singular value decomposition is

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

where $U \in \mathbb{R}^{m \times m}$ and $V \in R^{n \times n}$ are orthogonal matrices and $\Sigma \in R^{m \times n}$ is diagonal with nonnegative entries. The SVD is ubiquitous in numerical computing.

- Features of the decomposition
  - The SVD can provide an optimal low-rank approximation to a matrix.
  - Given an SVD you can form a pseudoinverse to your matrix.
- Example application areas
  - Data compression.
  - Signal processing.
  - Pattern recognition.
  - Certain least squares problems.

The Bidiagonalization Step

Given a matrix A, a bidiagonalization of A is a decomposition

$$A = UBV^T$$

where, again, $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices but the matrix $B \in \mathbb{R}^{m \times n}$ is bidiagonal.
What's the point:

- All further calculations can then be done on the sparse matrix B.
- Can be done stably using Householder reflectors.
- A finite time algorithm (old fashioned).

Householder Reflectors and the Bidiagonalization Algorithm

Left Householder

$$\begin{pmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \end{pmatrix} \rightarrow \begin{pmatrix} \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \end{pmatrix}$$

Right Householder

$$\begin{pmatrix} \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \\ 0 & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} & \tilde{x} \end{pmatrix} \rightarrow \left(\begin{array}{c|ccccc} \boldsymbol{x} & \boldsymbol{x} & 0 & 0 & 0 & 0 \\ \hline 0 & & & & & \\ 0 & & & & & \\ 0 & & & \widetilde{A} & & \\ 0 & & & & & \\ 0 & & & & & \end{array}\right)$$

Computational Considerations

- Features of the algorithm
  - Dense calculations.
  - The application of a Housholder reflector is independent for each column.
  - The size of the working set reduces with each step.
- Coding the algorithm for a GPU
  - Reduction: calculating vector norm to get the reflector
  - Both vector level and matrix level parallelization in applying the reflector (to do)

Formulation of Problem

We consider the following equation:

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

for $B \in \mathbb{R}^{n\times(n+1)}$ where

$$B = \begin{pmatrix} b_{11} & b_{21} & 0 & \cdots & 0 \\ 0 & b_{12} & b_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & b_{1n} & b_{2n} \end{pmatrix}$$

is a bidiagonal matrix.

- Our goal: To obtain the singular values
  (1) quickly, with
  (2) low memory usage

Reduction of Problem

We notice that this system can be reduced to solving similar subproblems, namely

$$B = \begin{pmatrix} B_1 & 0 \\ b_{1k}e_k & b_{2k}e_1 \\ 0 & B_2 \end{pmatrix}$$

where

(1) B_i are themselves bidiagonal matrices, and

(2) e_j denotes the standard unit vectors with 1 in the jth component

This is the fundamental design of any divide and conquer method. What makes this process different is that we maintain low memory usage by not retaining the singular vectors.

Formulation of Recursion

Given that

$$B_i = U_i \left(\Sigma_i \ 0\right) \left(V_i \ v_i\right)^T$$

are the decomposition of the sub matrices, it can be shown that

$$B = \tilde{U} \left(M \ 0\right) \left(\tilde{V} \ \tilde{v}\right)^T$$

where

$$M = \begin{pmatrix} r_0 & b_{1k} l_1 & b_{2k} f_2 \\ 0 & \Sigma_1 & 0 \\ 0 & 0 & \Sigma_2 \end{pmatrix}$$

where

(1) l_1 is the last row in V_1,

(2) f_2 the first row in V_2, and

(3) r_0 can be similarly derived from results of the lower levels.

Formulation of Recursion (cont'd)

For simplicity, let

- $z = (r_0 \;\; b_{1k} l_1 \;\; b_{2k} f_2)$
- $D_i = \operatorname{diag} d_i$
- $M = U_M \, (\Sigma \;\; 0) \, V_M$

Observe that

$$B = \tilde{U} U_M \, (\Sigma \;\; 0) \left(\tilde{V} V_M \;\; \tilde{v} \right)^T = U \, (\Sigma \;\; 0) \, (V \;\; v)^T$$

implies

- the singular values of M are the singular values of B
- the first row of $V = \tilde{f} V_M$ and the last row of $V = \tilde{l} V_M$ (and thus require $\mathcal{O}(n)$ space at any given time)

Conclusion of Recursion

This implies that by holding the

(1) singular values, and

(2) first and last rows

from lower levels, we can backsolve our desired singular values. (Assuming we can solve for Σ and V_M for a given M *cough*)

Solving for Σ, V_M

Given our z and $d = (d_1, d_2, \ldots, d_n)$, with $d_1 \equiv 0$, we observe

$$f(\sigma_i) = 1 + \sum_j \frac{z_j^2}{d_j^2 - \sigma_i^2} = 0$$

This is known as the secular equation.

- Observe (assuming we permute the system to order the d_i)

$$0 < \sigma_1 < d_2 < \sigma_2 < \ldots < \sigma_n$$

- Given our σ_i, there exists a solution to find the columns v_i of V_M.

Observation on Computation

- The singular values can be computed independently
- The individual components of $f = \tilde{f} V_M$ and $I = \tilde{I} V_M$ can be computed independently by calculating each column of V_M and doing the dot product.

This suggests the above calculations are amenable to OpenMP (and they are)

Finding Singular Vectors

- Given the singular values σ_i, right singular vectors are found using a "Twisted factorization", which is essentially an inverse power iteration.
- Left singular vectors are found from right singular vectors with a matrix multiply.
- Now we have the singular vectors for B, apply Householder reflections to get U and V for A.

Recall the inverse power iteration to find the eigenvector v of a matrix A corresponding to it's smallest eigenvalue. We pick an initial x^0 and then solve repeatedly

$$A\tilde{x}^{n+1} = x^n, \quad x^{n+1} = \frac{\tilde{x}^{n+1}}{||\tilde{x}^{n+1}||}$$

With shift we can find eigenvectors of the eigenvalue closest to some μ, in our case σ_i^2. We solve

$$(B^tB - \sigma_i^2 I)\tilde{x} = \gamma_k e_k$$

Twisted Factorization

- The twisted factoriation allows us to do an inverse power iteration accurately and quickly.
- First we compute the "forward" and "backward" factorizations for each singular value.

$$B^tB - \sigma_i^2 I = P(D1)P^t = Q(D2)Q^t$$

$$P = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ p_1 & 1 & 0 & \dots & \vdots \\ 0 & p_2 & 1 & \dots & \vdots \\ 0 & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & p_{m-1} & 1 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & q_1 & 0 & \dots & 0 \\ 0 & 1 & q_2 & \dots & \vdots \\ 0 & 0 & 1 & \ddots & 0 \\ 0 & \vdots & \ddots & \ddots & q_{m-1} \\ 0 & 0 & \dots & \dots & 1 \end{pmatrix}$$

$$D1 = \text{diag}(D1_1, D1_2, \dots, D1_m), \;\; D2 = \text{diag}(D2_1, D2_2, \dots, D2_m)$$

Twisted Factorization 2

Given the Forward and Backward factorizations of $B^tB - \sigma_i^2 I$, the Twisted factorization at k is

$$B^tB - \sigma_i^2 I = N_k D_k N_k^t$$

Where

$$N_k = \begin{pmatrix} 1 & & & & & & \\ p_1 & \ddots & & & & & \\ & \ddots & 1 & & & & \\ & & p_{k-1} & 1 & q_k & & \\ & & & & 1 & \ddots & \\ & & & & & \ddots & q_{m-1} \\ & & & & & & 1 \end{pmatrix}$$

$$D_k = \text{diag}(D1_1, \ldots, D1_{k-1}, \gamma_k, D2_{k+1}, \ldots, D2_m)$$

Where

$$\gamma_k = D1_k + D2_k - (\alpha_k^2 + \beta_{k-1}^2 - \sigma_i^2)$$

Solving the Twisted Factorization for singular vectors

- We solve the system $N_k D_k N_k^t \tilde{x} = \gamma_k e_k$ first, which through the black magic of linear algebra is equivalent to $N_k^t \tilde{x} = e_k$.
- For additional accuracy (needed specifically with clustered singular values), we do one more backsolve $N_k D_k N_k^t x = \tilde{x}$ to get our final singular value x for the reduced bidiagonal matrix.
- $y = \frac{Bx}{\sigma_i}$ gives our left singular vector for the reduced bidiagonal matrix.
- We still need to include the effects of the householder reflections from the bidiagonalization.

$$A = H_1 B H_2^t = H_1 (Y \Sigma X^t) H_2^t$$

- We apply the reflectors stored from the bidiagonalization part to the outputs X and Y.

Parallelization Ideas

This process is completely independent for each σ .

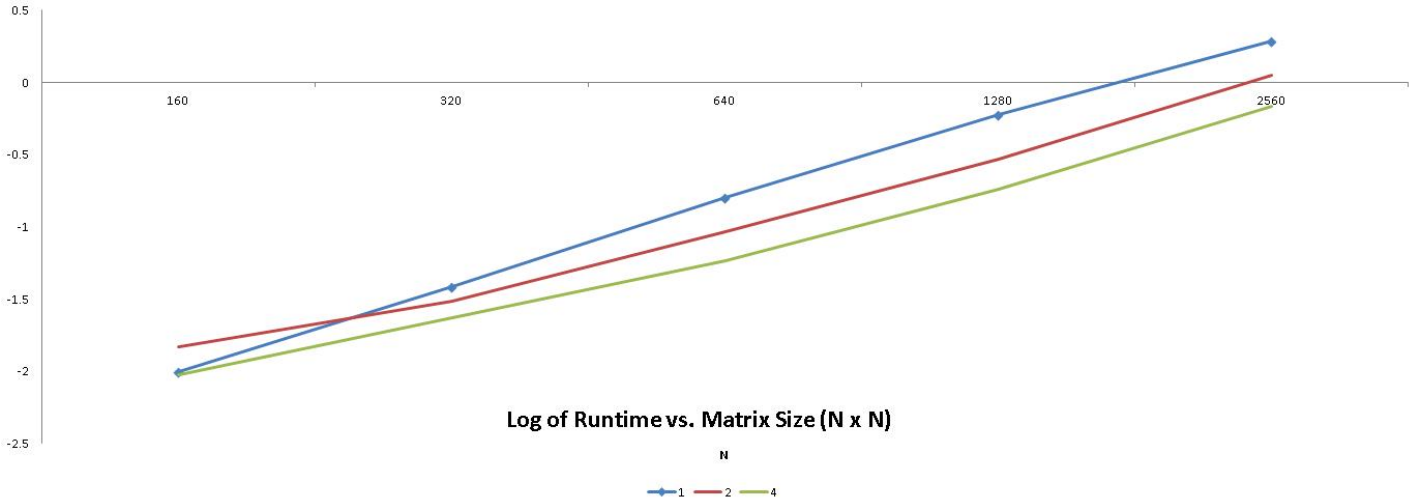
- Do the whole thing on the GPU
- ABANDONED IMMEDIATELY - lots of O(n) sequential calculations per work item. Things have to be stored in global, lots of global access.
- Can we at least do the Gamma Calculation on the GPU? $O(N^2)$ independent entries
- ABANDONED EVENTUALLY - way too much data transfer from the CPU to the device, $O(N^2)$, not THAT many flops.
- Ok, let's use OpenMP. This seemed to work.

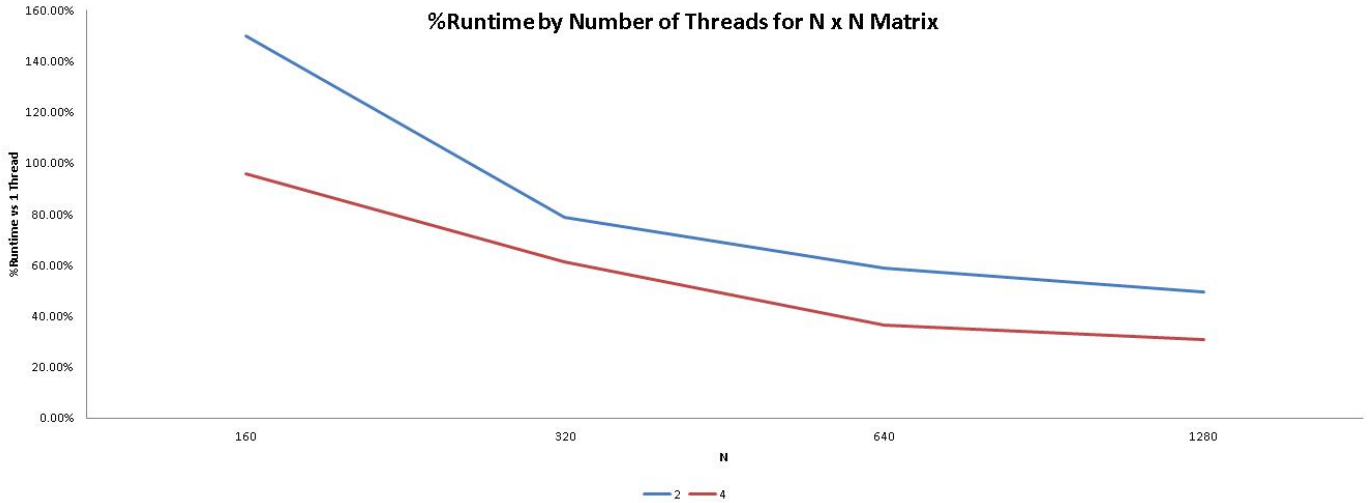
Additional Improvements

A few things can still be done to increase the speed of these procedures

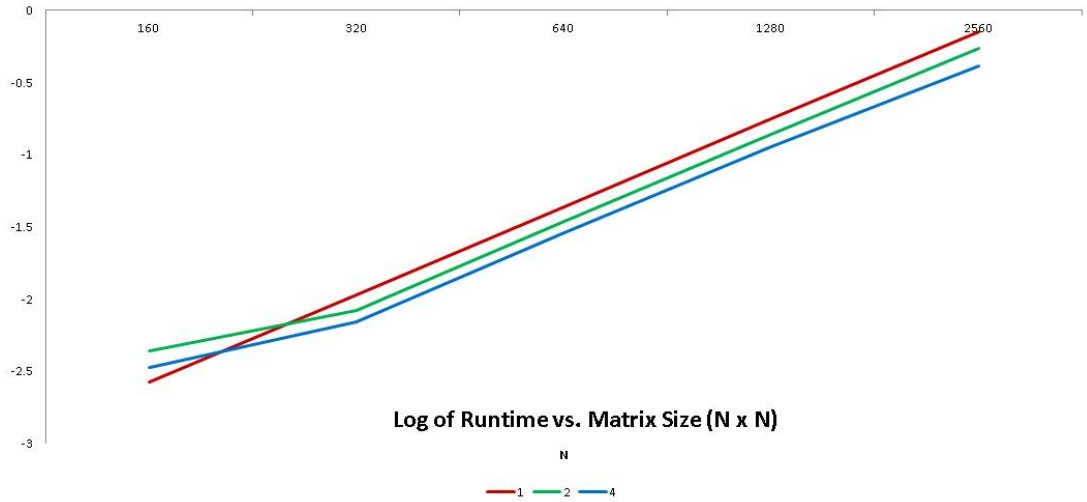
- Possibly moving the householder reflection application to the GPU would give some speedup. This is the slowest part (95% of the Singular Vector time for $1k \times 1k$), so improvements here would be worthwhile. However, we need to move $O(N^2)$ data to the device, which might kill it.
- The twisted factorization is done in pieces that work on all vectors, and paralellized within. This could be changed to do everything for an individual vector, and parallelize outside. Also could do less vectors if desired. This is a fast part anyway, so maybe the payoff isn't worth it. . .

Performance Graphs – Singular Values





Performance Graphs – Singular Vectors



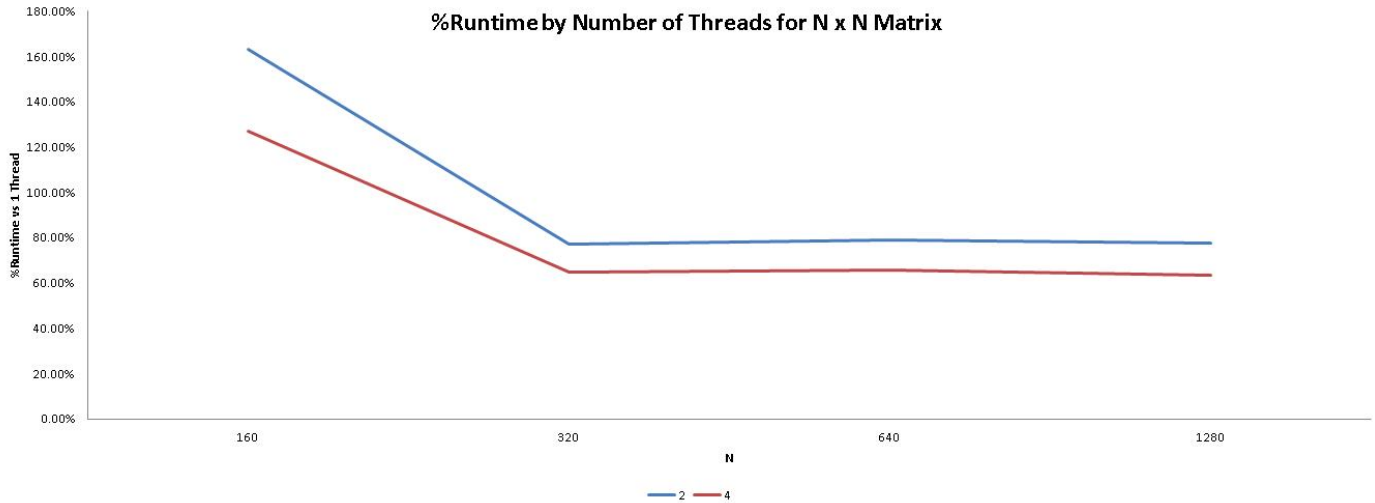


Image Compression Example