Randomized Sketching for Large-Scale Sparse Ridge Regression Problems

Chander Iyer [1] Christopher Carothers [1] Petros Drineas [2]

[1]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY
[2]Department of Computer Science, Purdue University, West Lafayette, IN

Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems
November 13, 2016

Outline

- Introduction.
  - "Big Data" in real time.
  - Randomization : An HPC perspective.
- Sparse least squares regression.
  - Regularized least squares solvers : exact v/s randomized.
  - Blendenpik : A randomized iterative least squares solver.
- Implementing Blendenpik on the Blue Gene/Q.
  - Distributed Blendenpik for large-scale sparse matrices.
  - Batchwise Blendenpik.
- Evaluation and Results.
  - Datasets.
  - Scalability analysis.
  - Performance analysis.
  - Numerical stability analysis.
- Summary and Future work.

"Big Data" in real time (Arjun Shankar, SOS17 Conference)

Social Medium	Data generation rate
	400M / day
	Images : 30B / month
	Mails : 419B / day
	Videos : 76PB / year

Table : Social Media data generation rate

	Sensor	Data generation rate
	Ion mobility spectroscopy	10TB / day
	Boeing Flight recorder	240TB / trip
	Astrophysics Data	10PB / year
	Square kilometer telescope array	480 PB / day

Table : Sensor data generation rate

Randomization : An HPC perspective

Numerical Algorithms and Libraries at Exascale, Dongarra et. al.,2015,`HPCwire`

- "...one of the most interesting developments in HPC math libraries is taking place at the intersection of numerical linear algebra and data analytics, where a new class of randomized algorithms is emerging...".
- "...powerful tools for solving both least squares and low-rank approximation problems, which are ubiquitous in large-scale data analytics and scientific computing."
- "these algorithms are playing a major role in the processing of the information that has previously lain fallow, or even been discarded, because meaningful analysis of it was simply infeasible-this is the so called 'Dark Data phenomenon'."

Randomized Algorithms (random sampling / random projections)

- Can be scaled with relative ease(!) compared to traditional solvers to modern HPC architectures.
- Numerically robust due to implicit regularization.

Least squares solvers

Regularized least squares Regression

$$y^* = \arg\min \|y\|_2 \text{ subject to } y \in \arg\min_x \|Ax - b\|_2^2 + \lambda \|x\|_2^2 \text{ where}$$

$$A \in \mathbb{R}^{m \times n}; \quad \mathbf{nnz}(A) \approx m * n; \quad m \gg n; \quad x \in \mathbb{R}^n.$$

Traditional ridge regression solvers are based on the solving in the dual space or using kernelized ridge regression that runs in $O(mn^2)$.

Randomized least squares solvers(Existing approaches)

- Sample rows after preprocessing A. Then apply QR on the sampled matrix.
  Drineas, Mahoney, Muthukrishnan & Sarlós, Numer. Math., 2011
- Construct a preconditioner from A. Then iteratively solve the preconditioned matrix.
  Rokhlin & Tygert, PNAS, 2008

Blendenpik(Avron, Maymounkov & Toledo, *SISC*, 2010)

- Combines both approaches that runs in $O(mn \log m)$ time.
- Preprocess A by applying a unitary transform. Then sample rows from this transform and apply QR to construct a preconditioner. Then iteratively solve the preconditioned matrix to construct an approximate solution.

The *Blendenpik* algorithm

Input: $A' \in \mathbb{R}^{(m+n)\times n}$ matrix, where $A' = \begin{pmatrix} A \\ \lambda I \end{pmatrix}$ $m \gg n$ and rank $(A) = n$.

$b' \in \mathbb{R}^{m+n}$ vector where $b' = \begin{pmatrix} b \\ 0 \end{pmatrix}$.

$F \in \mathbb{R}^{(m+n)\times(m+n)}$ random unitary transform matrix.

regularization parameter $\lambda > 0$ $\gamma(\geq 1)$ - Sampling factor.

Output: $\hat{x} =$ Solution of $\min_x \|Ax - b\|_2$.

while Output not returned **do**

$M = FA'$ — random unitary transformation

Let $\mathcal{S} \in \mathbb{R}^{(m+n)\times(m+n)}$ be a random diagonal matrix:

$$\mathcal{S}_{ii} = \begin{cases} 1 & \text{with probability } \frac{\gamma n}{m+n} \\ 0 & \text{with probability } 1 - \frac{\gamma n}{m+n} \end{cases}$$

$M_s = \mathcal{S}M$ — Sampling

$M_s = Q_s R_s$ — Thin QR preconditioning

$\hat{\kappa} = \kappa_{\text{estimate}}(R_s)$

if $\hat{\kappa}^{-1} > 5\epsilon_{\text{machine}}$ **then**

$y = \min_z \|A' R_s^{-1} z - b'\|_2$ — Preconditioned iterative solve

Solve $R_s \hat{x} = y$

return $\hat{x}$

else

if # iterations > 3 **then**

solve using Baseline Least squares and return

end if

Distributed Blendenpik for large-scale sparse matrices

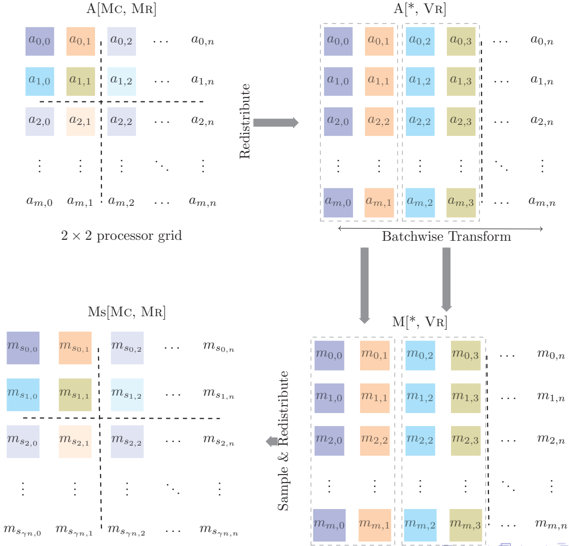
- The sparse unitary transformation is implemented using the Randomized Sparsity Preserving Transform (RSPT) proposed by Clarkson and Woordruff (runs in O(nnz(A)) time) and a combination of RSPT and 1-D routines of Discrete Cosine Transform(DCT) of the FFTW library.
- Distributed Blendenpik is implemented on top of Elemental. Elemental partitions the input matrices into rectangular process grids in a 2D cyclic distribution. The 2D input distribution format is locally non-contiguous, while the 1-D unitary transform needs locally contiguous columns on the input matrix. This redistribution is done by an `MPI_AlltoAll` collective operation.

Challenges

- **Memory Constraints:** The number of elements in a column is limited by the RAM available to the process assigned to that column. Also, a process may share the buffer with several columns at once.
- **MPI Framework Constraints:** The number of elements that can be redistributed in a collective operation is limited upto `INT_MAX`($2^{31} - 1$).

Batchwise Blendenpik

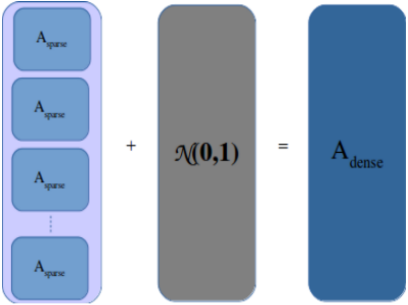
Solution Batchwise redistribution and transformation.



Datasets

Matrix Name	# of rows	# of columns	# of entries (Millions)	Condition number
ns3Da−8	163, 312	20, 414	16.77	$7.07E+002$
mesh deform−4	936, 092	9, 393	12.2	$1.17E+003$
memplus−32	568, 256	17, 758	13.26	$1.29E+005$
sls−1	1, 748, 122	62, 729	116.462	$8.67E+007$
rma10−8	374, 680	46, 835	36.18	$7.98E+010$
c-41−32	312, 608	9, 769	6.3	$4.78E+012$

Table : Matrices used in our evaluations.



Evaluation metrics

Let $A' \in \mathbb{R}^{(m+n)\times n}$ be the input matrix, $b' \in \mathbb{R}^{(m+n)}$ be the right hand side vector and let:

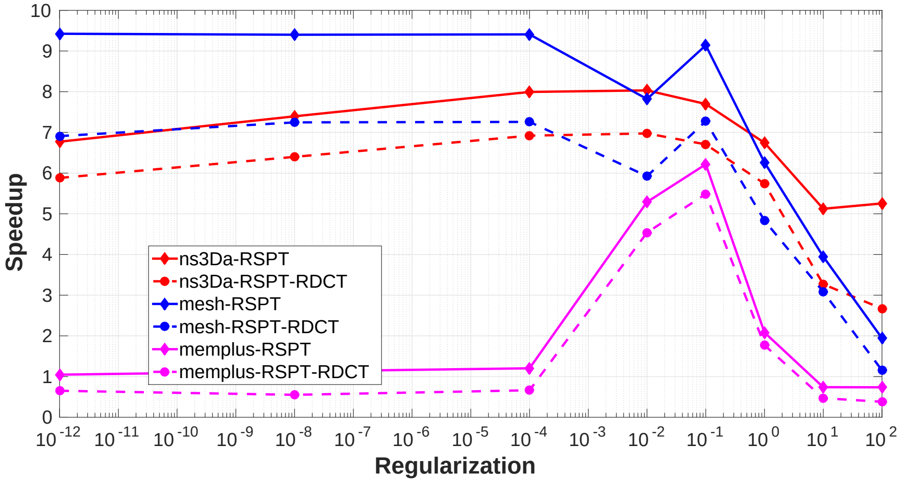
$$
\begin{aligned}
\hat{x} &\longleftarrow \text{ the min-norm solution obtained from batchwise Blendenpik} \\
x^* &\longleftarrow \text{ the exact solution} \\
\hat{r} &\longleftarrow \text{ the residual error, defined as } b' - A'\hat{x}. \\
\hat{t}_{\text{run}} &\longleftarrow \text{ running time of Blendenpik.} \\
t^*_{\text{run}} &\longleftarrow \text{ running time of baseline (Elemental).}
\end{aligned}
$$

We evaluate the Blendenpik algorithm using the following metrics.

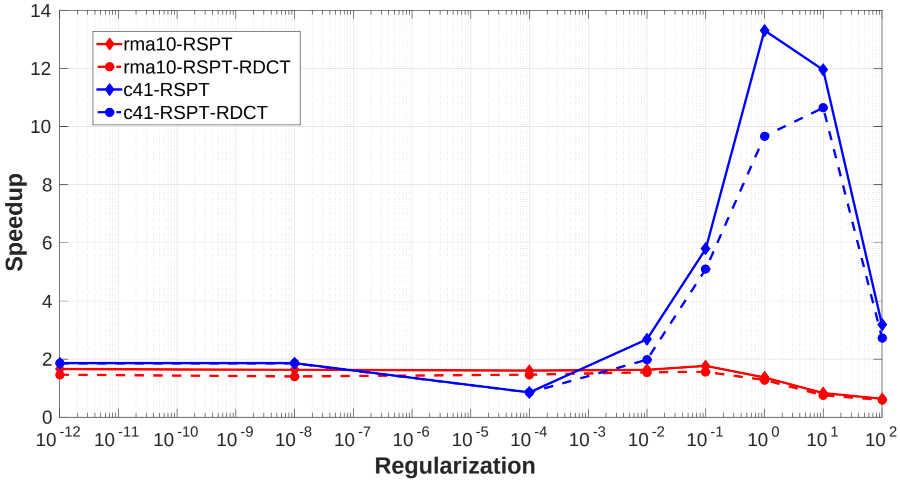
Speedup : given by $\frac{t^*_{\text{run}}}{\hat{t}_{\text{run}}}$.

Accuracy : defined in terms of the relative error for the min-norm solution $\hat{x}$ given by $\frac{\|A'\hat{x} - A'x^*\|_2}{\|A'x^*\|_2}$ and the backward error given by $\|A'^T\hat{r}\|_2$.

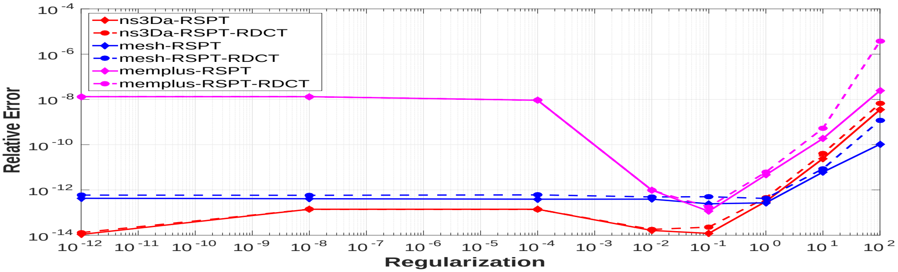
Speedup analysis for well-conditioned matrices for increasing regularization values.

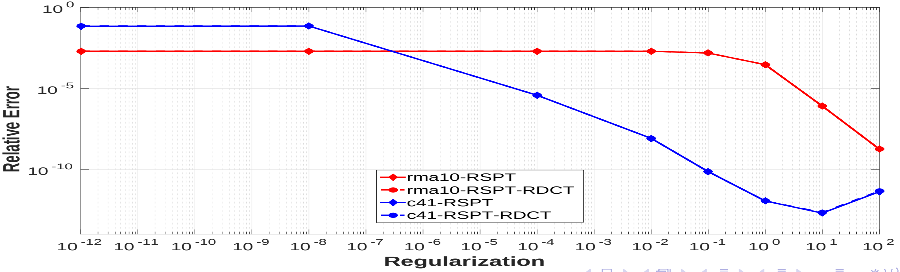


Speedup analysis for ill-conditioned matrices for increasing regularization values.

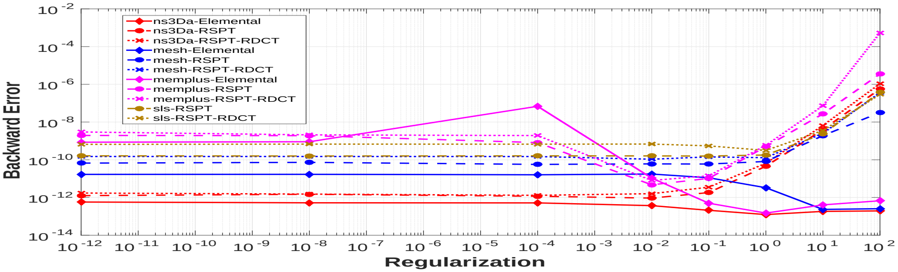


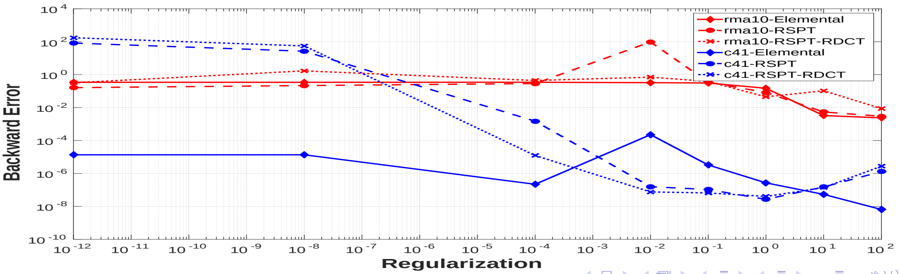
Relative Error as a function of λ.



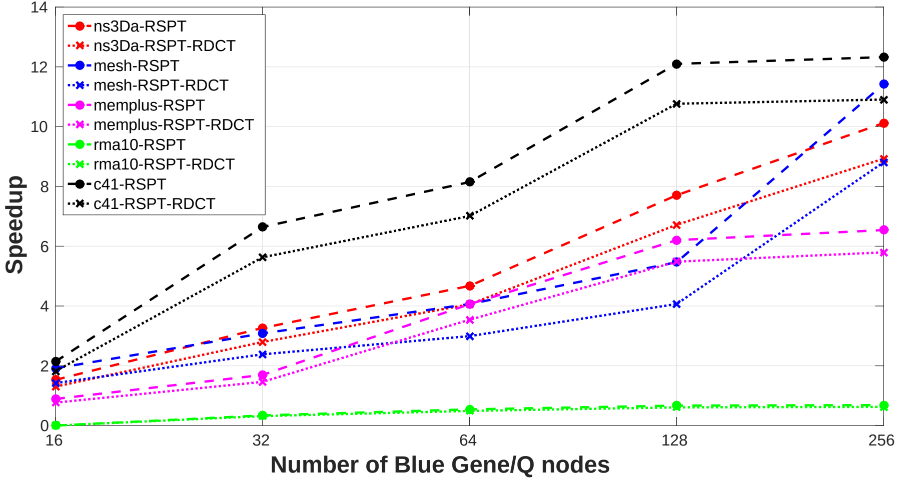


Backward Error as a function of λ.

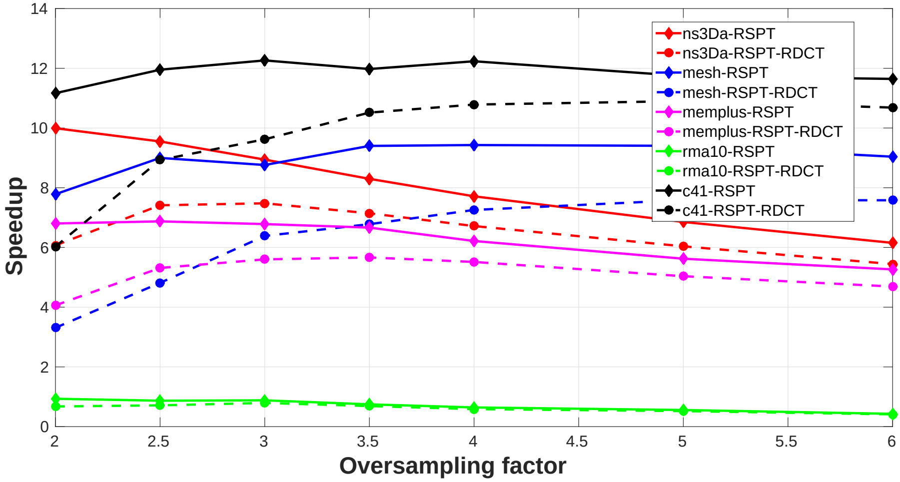




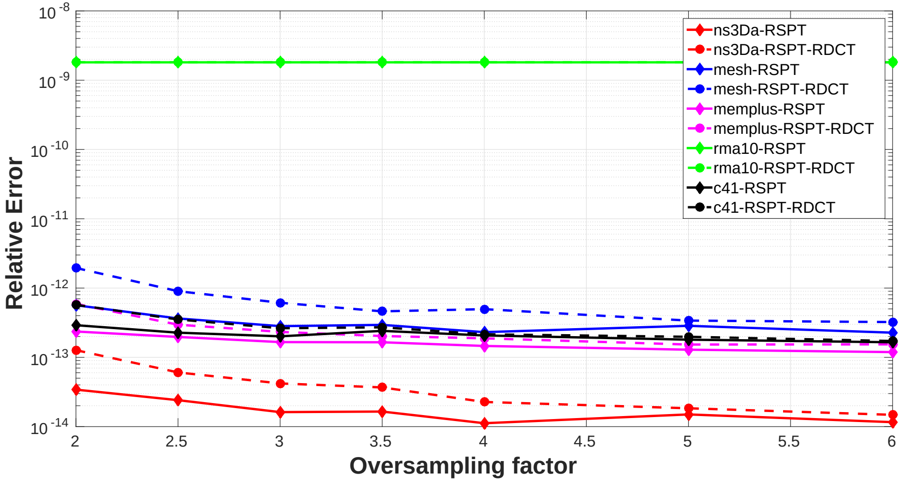
Strong Scaling as a function of increasing Blue Gene/Q nodes at optimal regularization value λ^*.



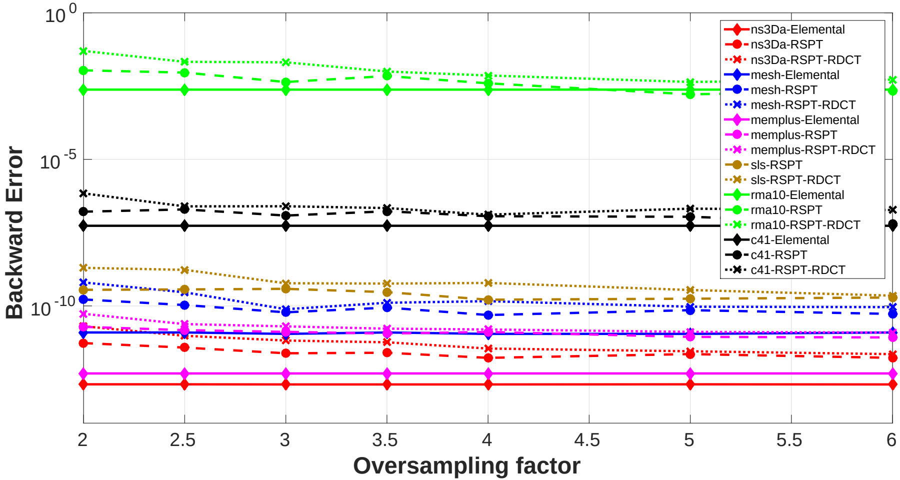
Speedup as a function of increasing oversampling factors at optimal regularization value λ^*.



Relative Error as a function of increasing oversampling factors at optimal regularization value λ^*.



Backward Error as a function of increasing oversampling factors at optimal regularization value λ^*.



Summary

- The speedup achieved by RSPT is always better than the RSPT-RDCT transform for two reasons. First, the Blendenpik algorithm spends reasonable time to compute the RDCT transform. Second, the RSPT produces a better preconditioner than the RSPT-RDCT transform that leads to faster convergence of the LSQR stage.
- As the regularization values increase, the speedup increases until it peaks for a certain regularization value and then reduces again for all matrices with the exception of the `rma10-8` matrix.
- The relative error decreases with increasing values of the regularization parameter until it achieves the smallest relative error at $\lambda = \lambda^*$ chosen as the optimal regularizer for our evaluations. As the condition numbers of the matrices increase, λ^* for each matrix also increases.
- The sparse randomized transforms demonstrate significant strong scaling for all matrices at λ^* with the exception of the `rma10-8` matrix.
- The Blendenpik solver demonstrates excellent speedup and numerical stability in terms of the relative error at λ^* for increasing oversampling factors. The backward error is somewhat worse yet comparable to the backward error achieved by the baseline sparse Elemental solver at λ^*.

Future Work

- **Sparse QR preconditioning** The most inhibitive stage of the Blendenpik algorithm is the dense QR preconditioning stage which for a sketched matrix $M_s \in \mathbb{R}^{\gamma n \times n}$ runs in $O(n^3)$ time. For sparse approximately-square matrices applying a sparse random transform results in a sparse sketch which makes a dense QR preconditioner an unsuitable choice. In such a scenario, Sparse QR based on multifrontal QR factorization is a suitable choice for this stage due to its $O(n^2)$ runtime.
- **Restarted LSQR** A common problem with certain matrices that have heavy-tailed singular spectra is that even though the preconditioner constructed in well-conditioned, the LSQR stage for such matrices converge extremely slowly leading to stagnancy. One way to resolve stagnancy is using a Restarted LSQR solver similar to the Bidiagonal Block Lanczos approach in Algorithm.
- **Ridge Regression variants** Another key improvement for the sparse ridge-regression problem that we envision is extending the Blendenpik algorithm framework to include other ridge-regression variants like dual ridge regression and kernel ridge regression.

Thank you !!!