A Scalable Randomized Least Squares Solver for Dense Overdetermined Systems

Chander Iyer [1] Haim Avron [2] Georgios Kollias [3]
Yves Ineichen [4] Christopher Carothers [1] Petros Drineas [1]

[1]Rensselaer Polytechnic Institute
[2]Tel Aviv University
[3]IBM Research - T. J. Watson Research Center
[4]IBM Research - Zurich Research Lab

Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems
November 16, 2015

Outline

- Introduction.
  - "Big Data" in real time.
  - Randomization : An HPC perspective.
- Dense least squares regression.
  - Least squares solvers : exact v/s randomized.
  - Blendenpik : A randomized iterative least squares solver.
- Implementing Blendenpik on the Blue Gene/Q.
  - Distributed Blendenpik for terabyte 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 (**Caveat!**).

Least squares solvers

Dense least squares Regression

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

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

Traditional non-iterative solvers and based on the classical QR algorithm that runs in $O(mn^2)$ and may be computationally expensive.

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: $A \in \mathbb{R}^{m \times n}$ matrix, $m \gg n$ and rank $(A) = n$.
$b \in \mathbb{R}^m$ vector.
$F \in \mathbb{R}^{m \times m}$ random unitary transform matrix.
$\gamma (\geq 1)$ - Sampling factor.
Output: $\hat{x} =$ Solution of $\min_x \|Ax - b\|_2$.
while Output not returned **do**
  $M = FA$ — random unitary transformation
  Let $\mathcal{S} \in \mathbb{R}^{m \times m}$ be a random diagonal matrix:

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

  $M_s = \mathcal{S} M$ — Sampling
  $M_s = Q_s R_s$ — Thin QR preconditioning
  $\hat{\kappa} = \kappa_{\mathrm{estimate}}(R_s)$
  if $\hat{\kappa}^{-1} > 5\epsilon_{\mathrm{machine}}$ **then**
    $y = \min_z \|AR_s^{-1} z - b\|_2$ — 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
  end if
end while

Distributed Blendenpik for terascale matrices

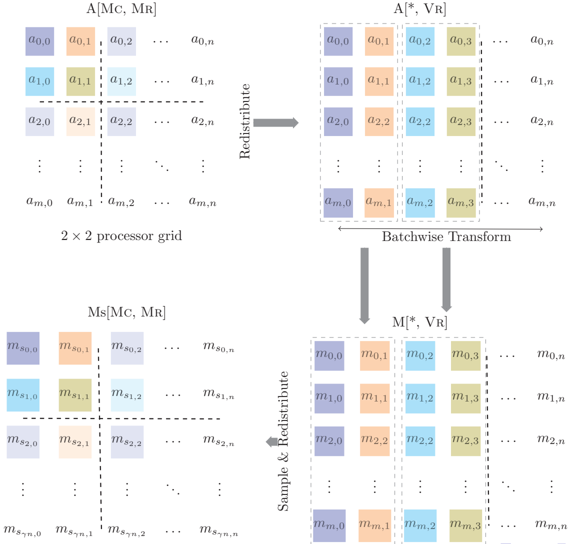
- Distributed Blendenpik is implemented on top of Elemental. Elemental partitions the input matrices into rectangular process grids in a 2D cyclic distribution.
- The unitary transformation is implemented using the 1-D routines of Discrete Cosine Transform(DCT) of the FFTW library.
- The 2D input distribution 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

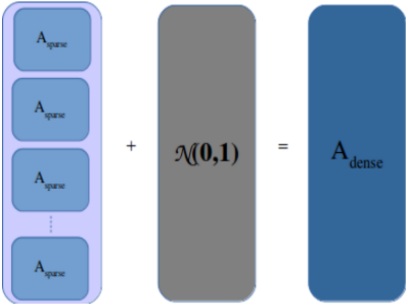
Solution Batchwise redistribution and transformation.



Datasets

Data Set	Number of rows	Number of columns	Number of Non zeros
Yoshiyasu Mesh	234023	9393	853829
ESOC Springer	327062	37830	6019939
Rucci	1977885	109900	7791168

Table : Sparse base datasets used in data replication



Data Set	Maximum number of replicated rows (Million)	Total number of entries (Billion)	Total size (TB)
Yoshiyasu Mesh	$\sim$ 44.932	422.050	3.070
ESOC Springer	$\sim$ 20.931	791.856	5.761
Rucci	$\sim$ 5.933	652.108	4.744

Table : Maximum dataset sizes used in Blendenpik evaluation

Evaluation metrics

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

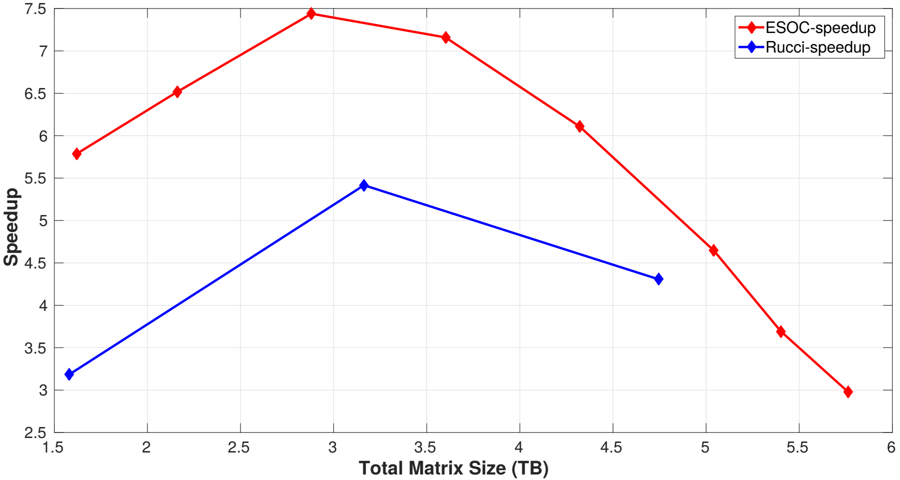
$$\begin{aligned} \hat{x} &\longleftarrow \text{ the min-norm solution obtained from batchwise Blendenpik} \\ x^{*} &\longleftarrow \text{ the exact solution} \\ \hat{r} &\longleftarrow \text{ the residual error, defined as } b - A\hat{x}. \\ \hat{t}_{\text{run}} &\longleftarrow \text{ running time of Blendenpik.} \\ t^{*}_{\text{run}} &\longleftarrow \text{ running time of baseline (Elemental).} \end{aligned}$$

We evaluate the Blendenpik algorithm using the following metrics.

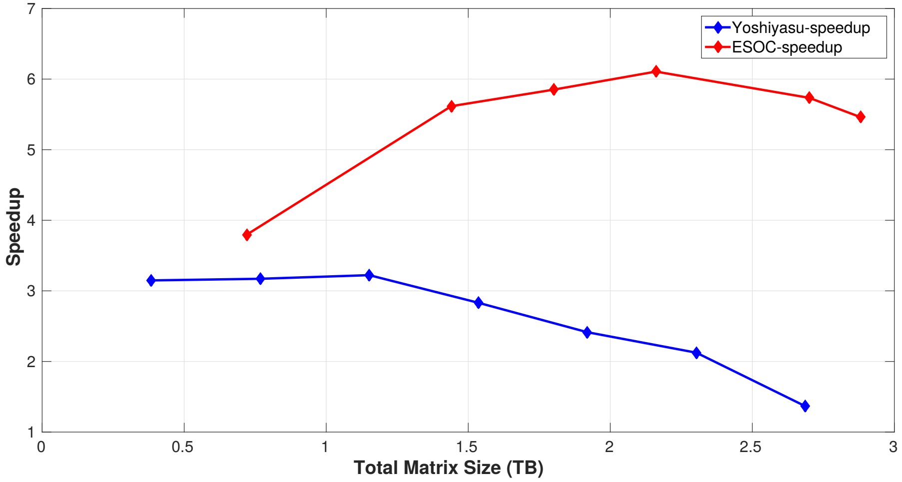
Speedup : given by $\dfrac{t^{*}_{\text{run}}}{\hat{t}_{\text{run}}}$.

Accuracy : defined in terms of the relative error for the min-norm solution $\hat{x}$ given by $\dfrac{\|A\hat{x} - Ax^{*}\|_2}{\|Ax^{*}\|_2}$ and the backward error given by $\|A^{T}\hat{r}\|_2$.

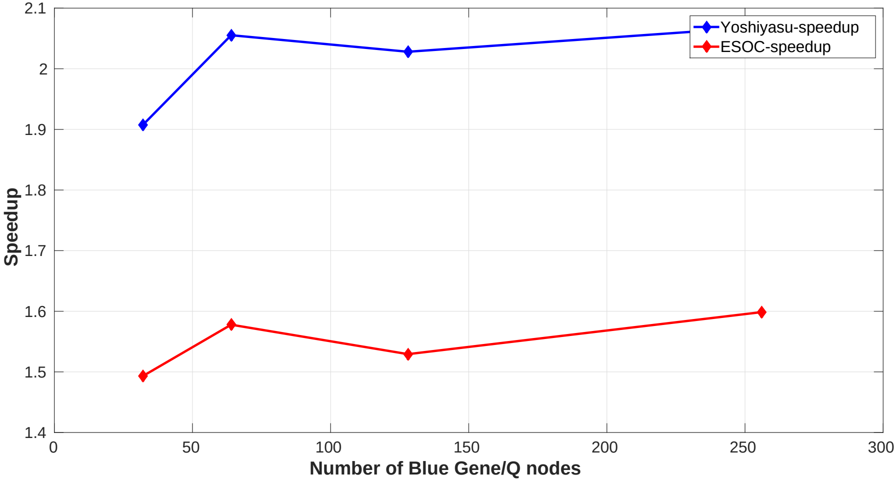
Speedup analysis for ESOC Springer and Rucci dense matrices for 1024 BG/Q nodes.



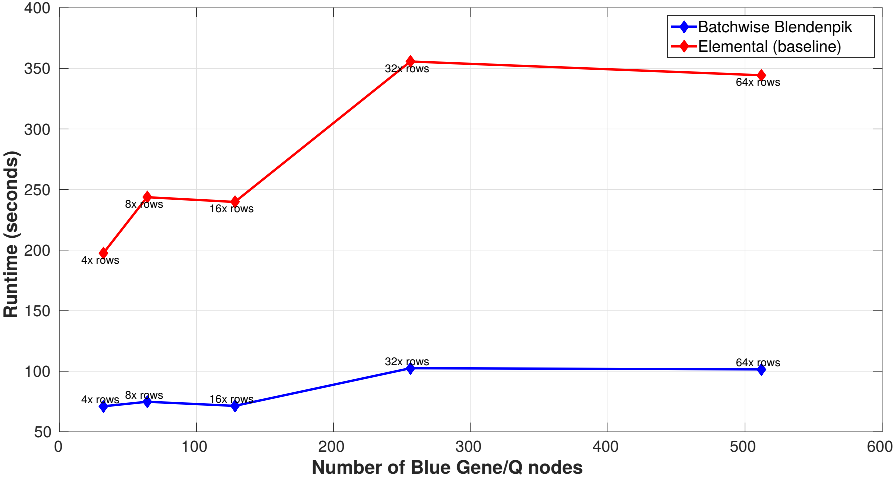
Speedup analysis for Yoshiyasu Mesh and ESOC Springer dense matrices for 512 BG/Q nodes.



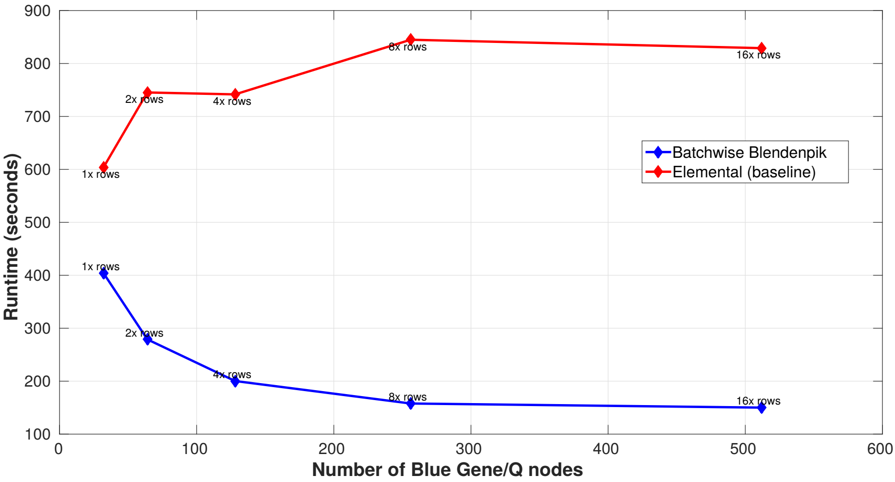
Strong scaling speedup analysis for the Yoshiyasu Mesh matrix (234023×9393) and ESOC Springer matrix (327062×37830) for increasing Blue Gene/Q nodes.



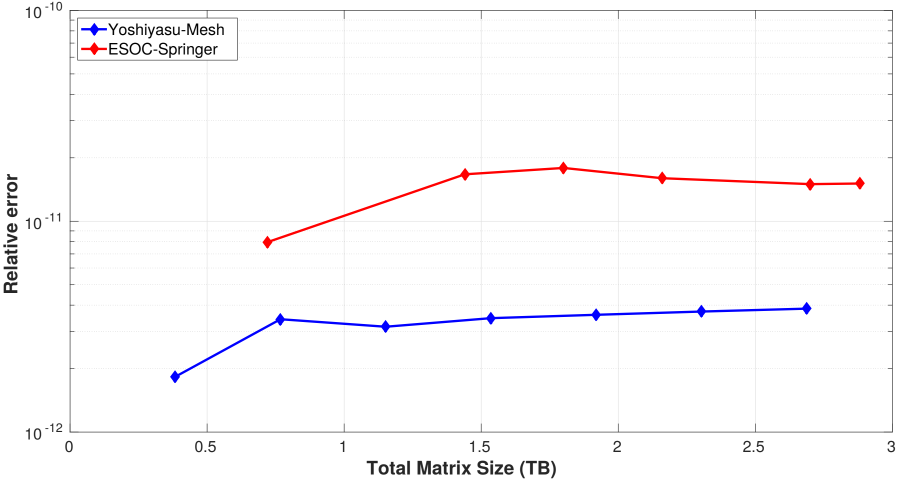
Weak scaling runtime analysis for the Yoshiyasu Mesh matrix (234023 × 9393) for increasing matrix sizes and increasing Blue Gene/Q nodes.



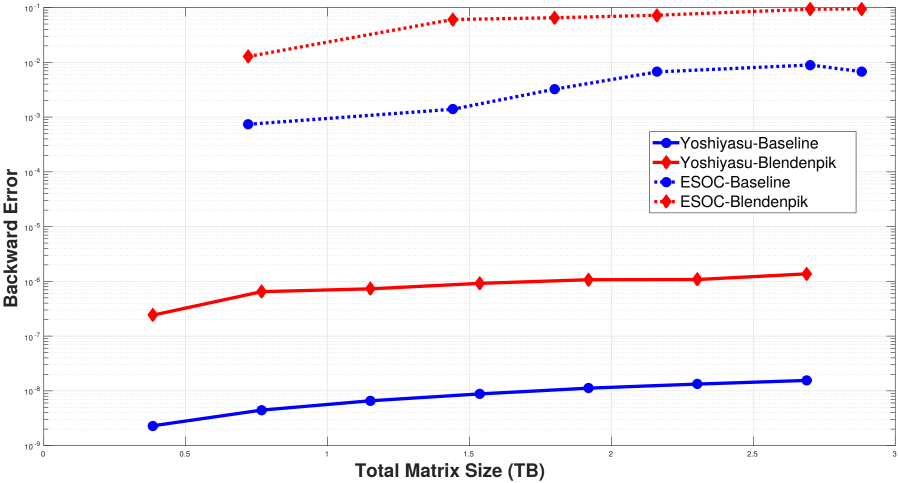
Weak scaling speedup analysis for the ESOC Springer matrix (327062 × 37830) for increasing matrix sizes and increasing Blue Gene/Q nodes.



Accuracy analysis in terms of relative error as a function of increasing matrix size for Yoshiyasu Mesh and ESOC Springer matrices for 512 BG/Q nodes.



Accuracy analysis in terms of backward error as a function of increasing matrix size for Yoshiyasu Mesh and ESOC Springer matrices for 512 BG/Q nodes.



Summary and Future Work

Summary

- The scalability of batchwise Blendenpik is determined by the number of columns in each batch of the DCT transform which in turn is determined by the number of rows of the matrix.
- The batchwise Blendenpik solver demonstrates appreciable strong scaling and weak scaling comparable to the baseline Elemental solver.
- The solver demonstrates excellent numerical stability in terms of the relative error. The backward error however is worse, though this is comparable to the backward error achieved by the baseline Elemental solver.

Future Work

- Perform unitary transformation only after an initial reduction of row space using input-sparsity sketching, as suggested by Clarkson and Woodruff(*STOC*,2013). This also helps us to choose a larger sample size for the preconditioning stage that can lead to a significant improvement in the numerical stability.
- Design a more finely tuned Blendenpik-based algorithm by reducing the communication overhead involved.

Thank you !!!