Communication-Avoiding Methods for Regularized Least-Squares

Aditya Devarakonda

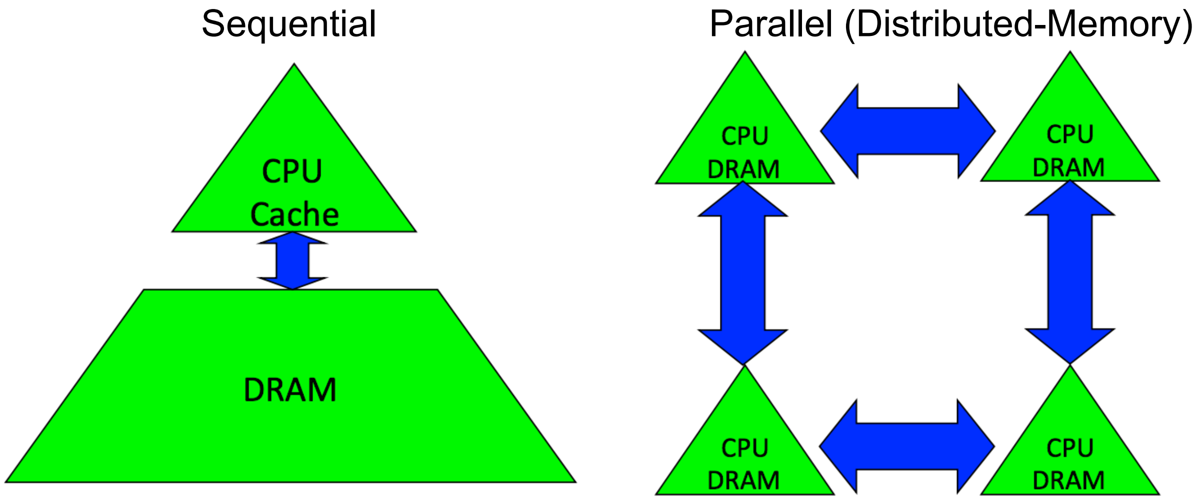
SIAM Annual

July 14th, 2017

Collaborators:
Jim Demmel (Adviser), Kimon Fountoulakis (Post-Doc), and Michael W. Mahoney (Co-adviser)

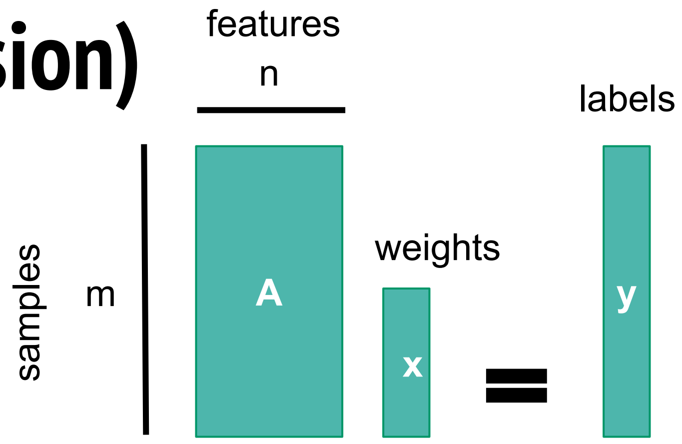
Definition

Communication is data movement.



Courtesy: Demmel

Least-Squares (Linear Regression)



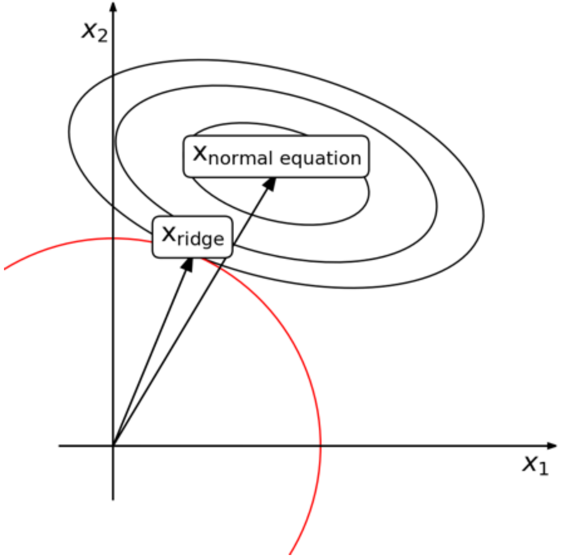
Many ways to solve.

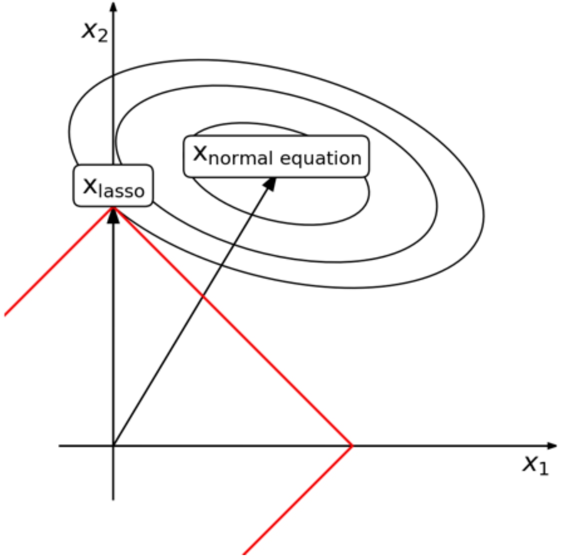
- <u>Direct</u>
  - Explicitly solve normal equation.
  - Implicitly through matrix factorizations.
- <u>Iterative</u>
  - Krylov methods (e.g. Conjugate Gradients).
  - **(Block) Coordinate Descent.**

Least-Squares: $\operatorname{argmin}_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - y\|_2^2$

Normal Equation: $x = (A^T A)^{-1} A^T y$

Regularized Least-Squares (Regression)



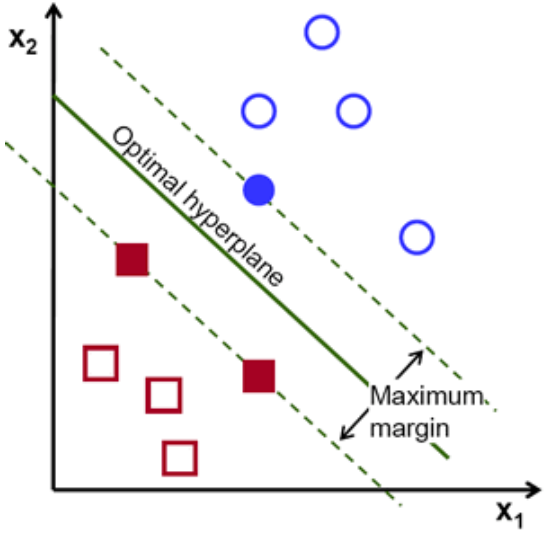


Ridge: $\operatorname{argmin}_{x\in\mathbb{R}^n} \underbrace{\frac{1}{2}\|Ax - y\|_2^2}_{\text{Loss function.}} + \underbrace{\frac{\lambda}{2}\|x\|_2^2}_{\text{Regularization function.}}$

LASSO: $\operatorname{argmin}_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax - y\|_2^2 + \boxed{\lambda\|x\|_1}$

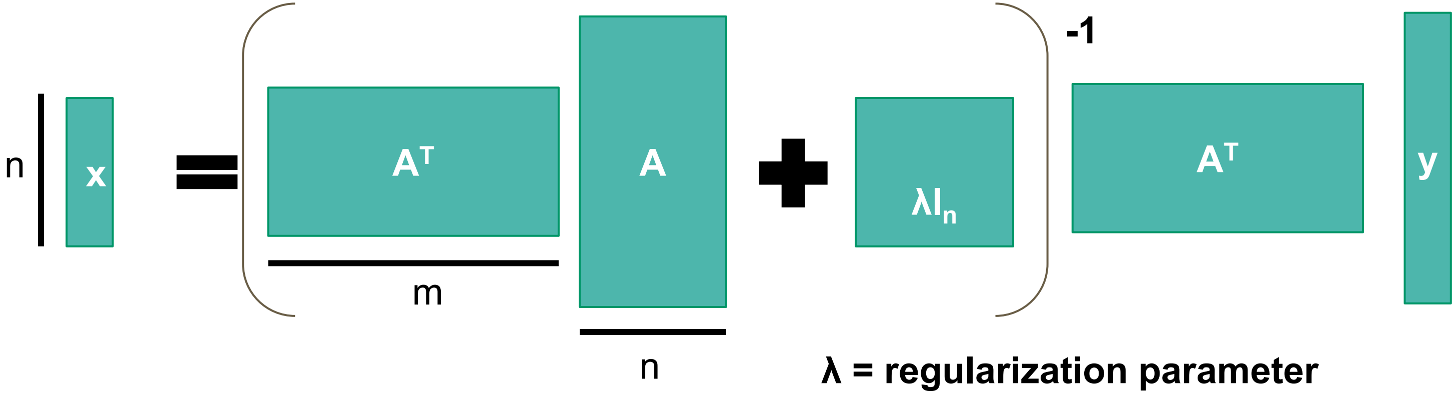
Binary Classification

Support Vector Machines



Ridge Regression with Block Coordinate Descent

Ridge solution:
(closed-form)



λ = regularization parameter
I_n = n x n Identity matrix
y = labels

Similar to normal equation.

Ridge Regression with Block Coordinate Descent

Ridge solution:
(closed-form)

$$x = \left(A^T A + \lambda I_n \right)^{-1} A^T y$$

Block Coordinate Descent

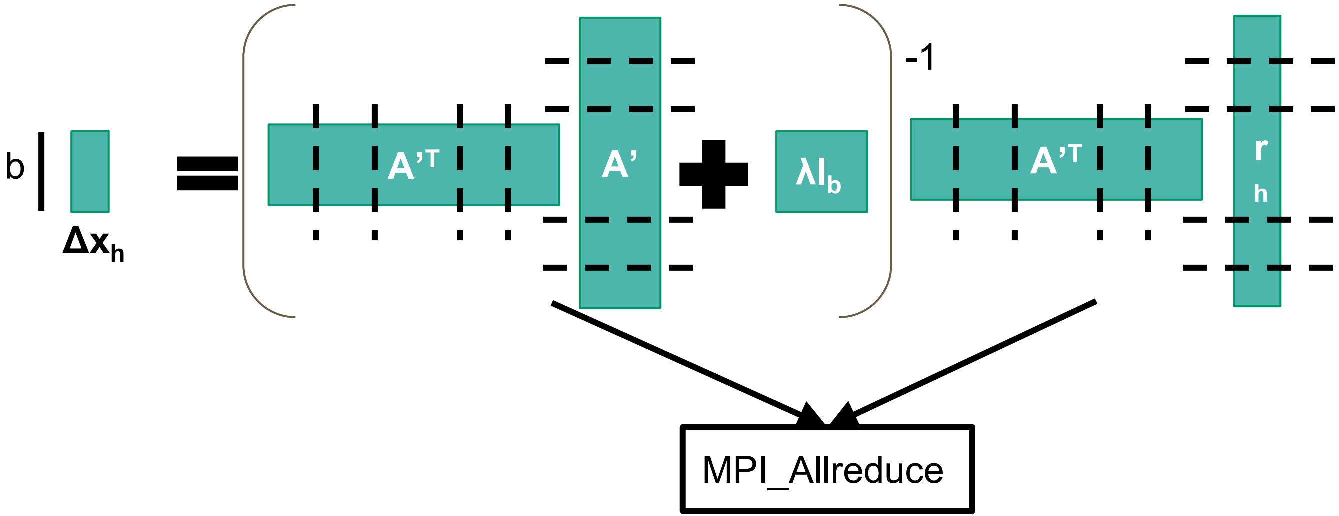
λ = regularization parameter
I_b = b x b Identity matrix
r = residual

A' = subsampled columns.

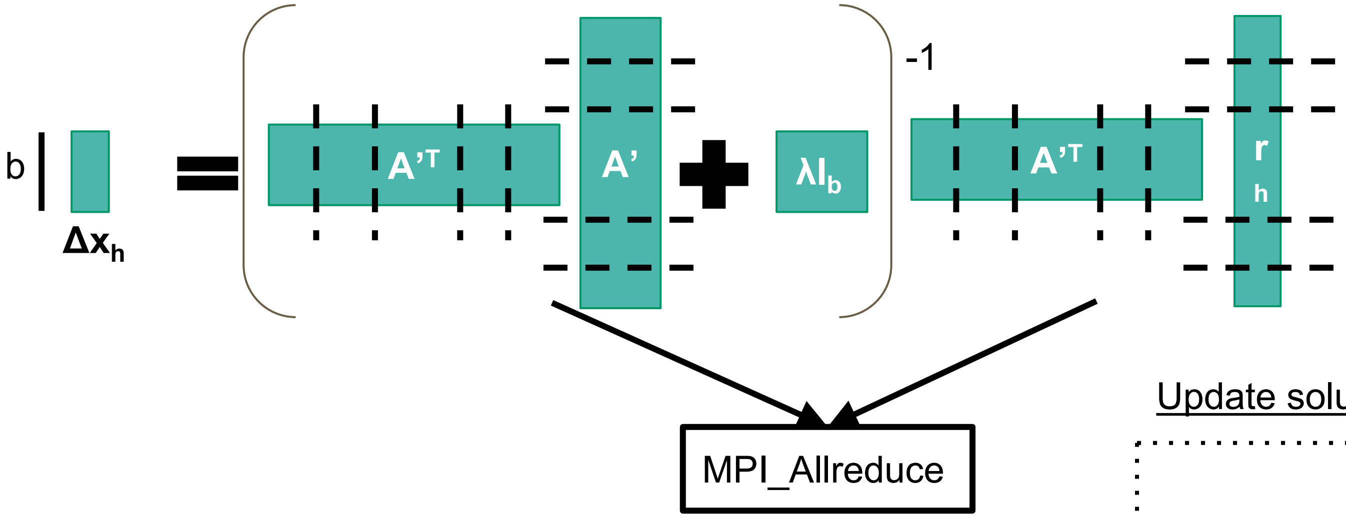
$$\Delta x_h = \left(\underbrace{A'^T}_{m} \underbrace{A'}_{b} + \lambda I_b \right)^{-1} A'^T r$$

Solution to sub-problem.

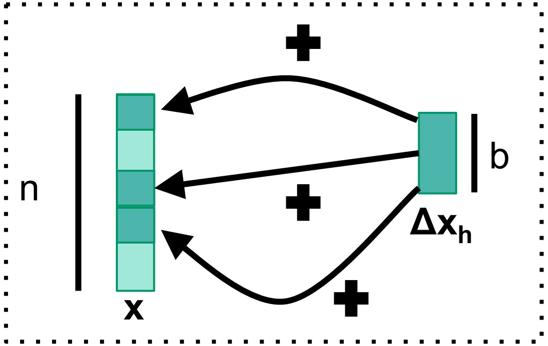
Block Coordinate Descent in Parallel



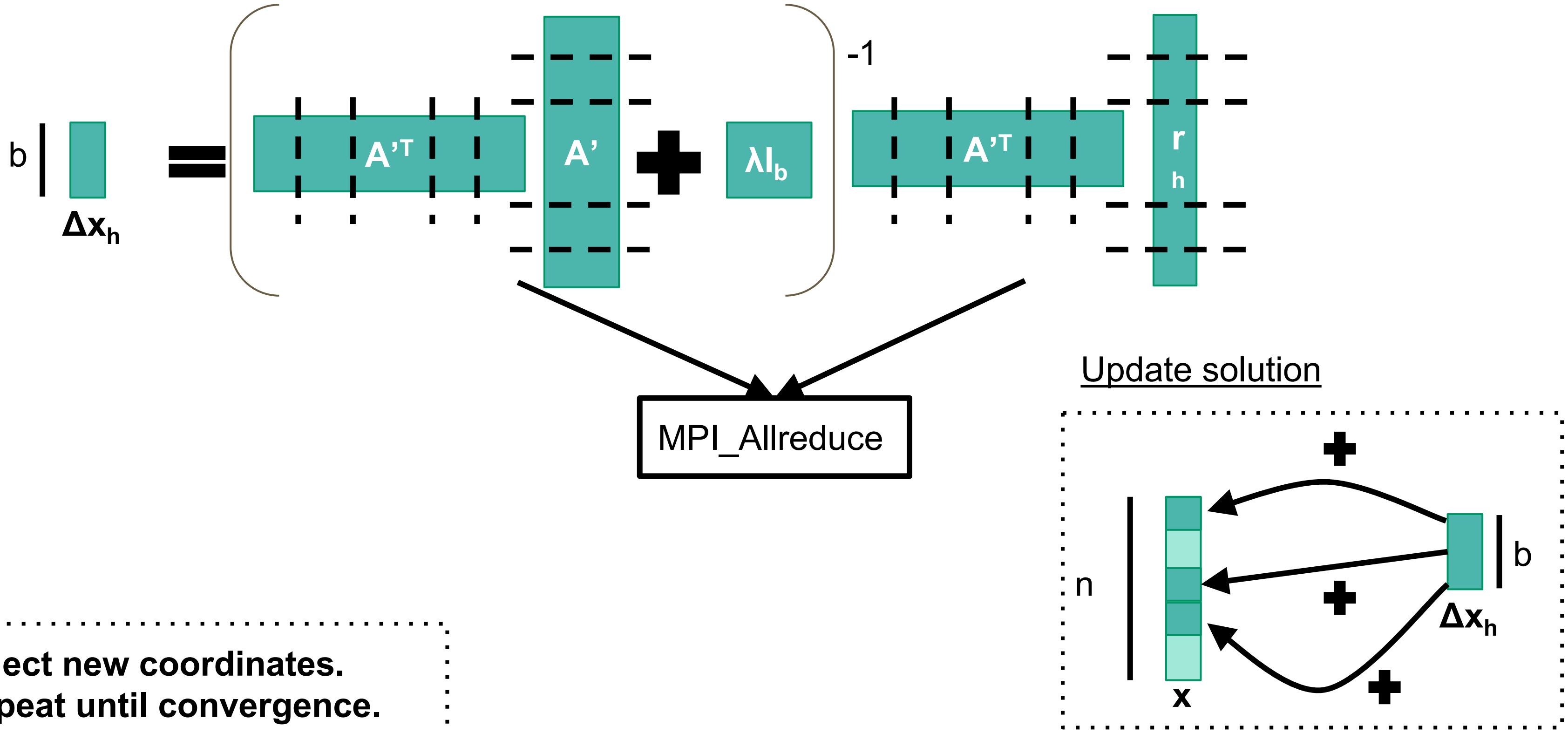
Block Coordinate Descent in Parallel



Update solution

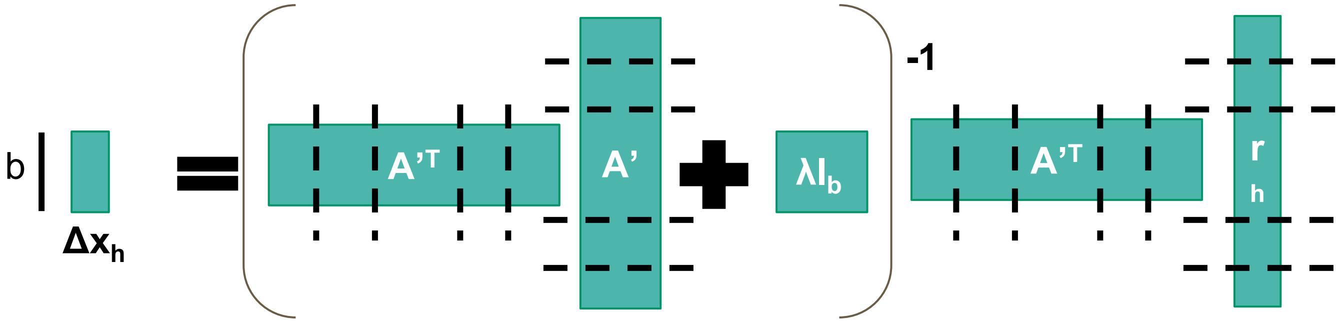


Block Coordinate Descent in Parallel



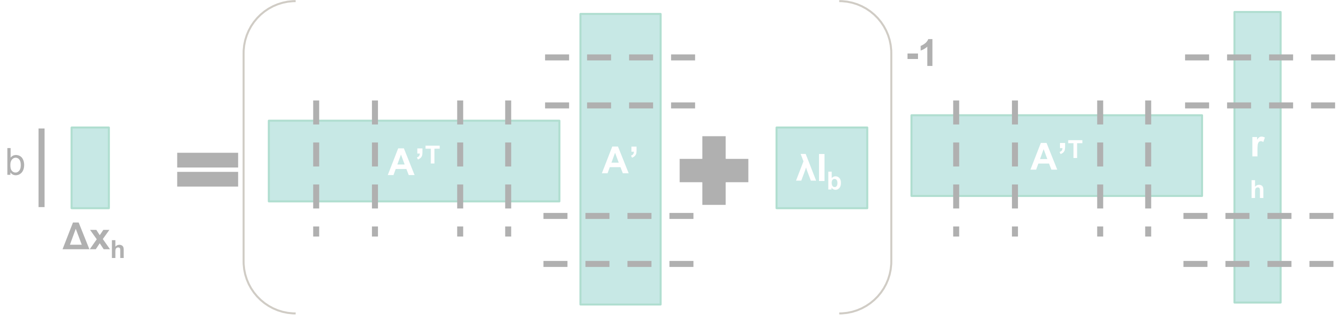
Select new coordinates.
Repeat until convergence.

Communication-Avoiding Block Coordinate Descent

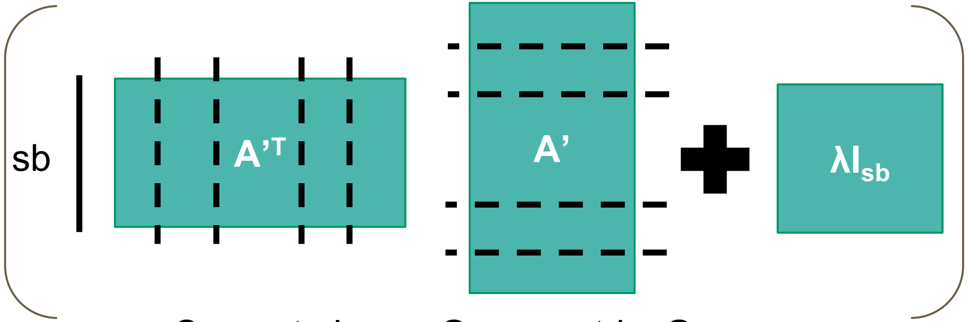


Block Coordinate Descent

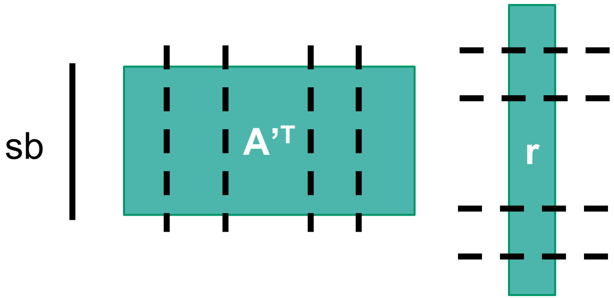
Communication-Avoiding Block Coordinate Descent



s = CA parameter
I_{sb} = sb x sb Identity
r = residual



Compute larger Gram matrix, **G**.
(Do not invert!)

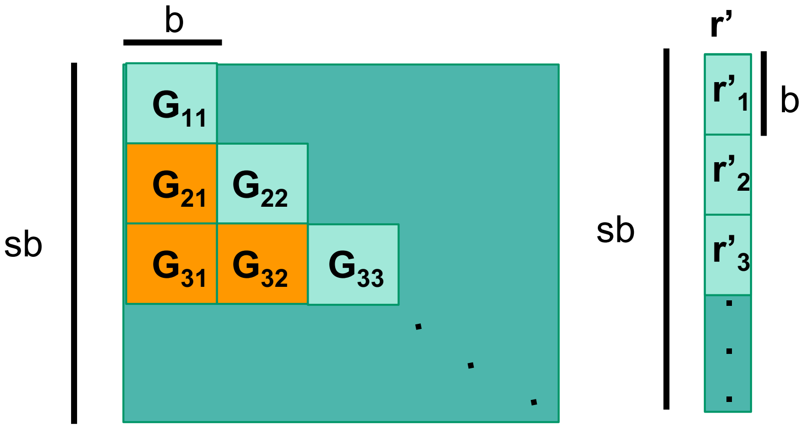


Larger residual contribution.
r is from this iteration

Redundantly on all processors.

Dimensions exaggerated for clarity.

Communication-Avoiding Block Coordinate Descent



Partition G and r'.

(**G** is symmetric)

Redundantly on all processors.

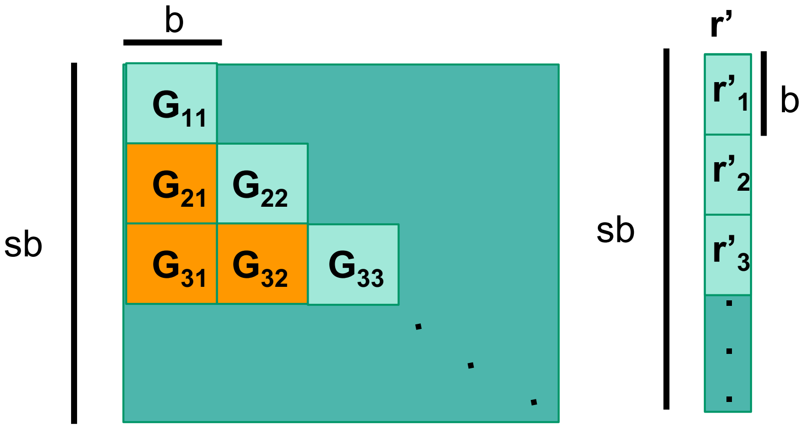
No communication for inner iterations.

Communication-Avoiding Block Coordinate Descent

Solution to 1st sub-problem.

Inner iteration 1

$$\Delta x_1 = \left(G_{11} \right)^{-1} r'_1$$



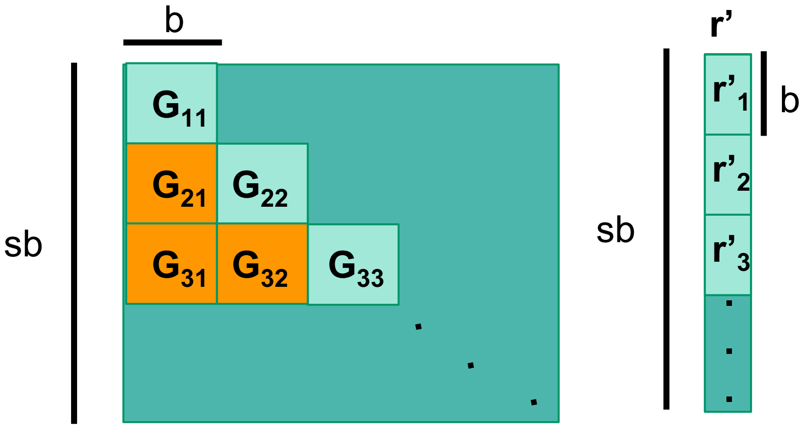
r_2', r_3',... become stale as soon as Δx_1 is computed.

Orange blocks are the updates

Redundantly on all processors.

No communication for inner iterations.

Communication-Avoiding Block Coordinate Descent



Inner iteration 1

$$\Delta x_1 = \left(G_{11}\right)^{-1} r'_1$$

Inner iteration 2

$$\Delta x_2 = \left(G_{22}\right)^{-1} r'_2 - G_{21}\,\Delta x_1$$

Solution to 2nd sub-problem.

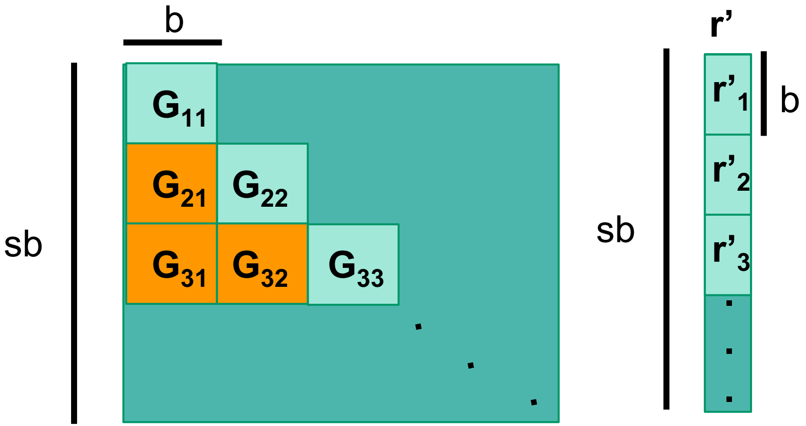
r_2', r_3',... become stale as soon as **Δx_1** is computed.

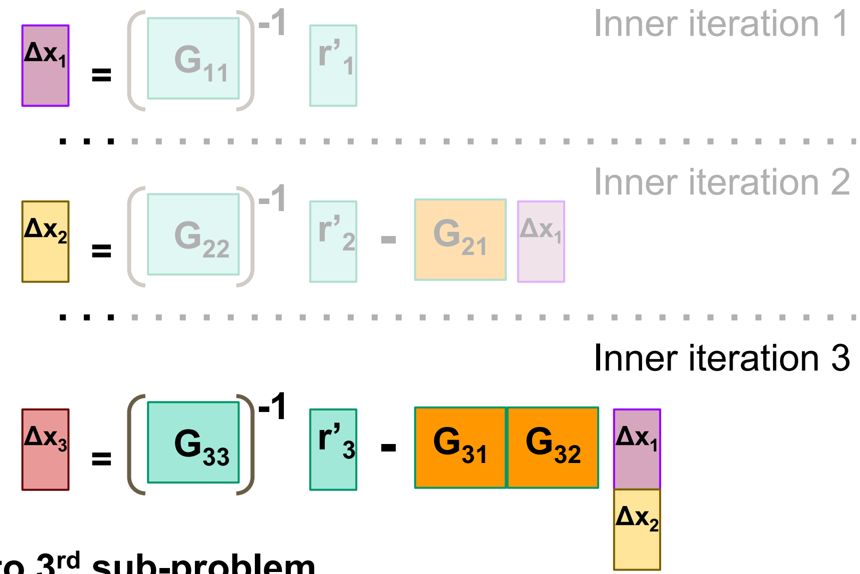
Orange blocks are the updates

Redundantly on all processors.

No communication for inner iterations.

Communication-Avoiding Block Coordinate Descent





Solution to 3rd sub-problem.

Subtraction terms are needed for CA and non-CA solutions to agree.

Algorithm 1 Block Coordinate Descent (BCD) Algorithm

Input: $X \in \mathbb{R}^{d \times n}, y \in \mathbb{R}^n,\ H > 1,\ w_0 \in \mathbb{R}^d,\ b \in \mathbb{Z}_+$ s.t. $b \leq d$
for $h = 1, 2, \cdots, H$ **do**
  choose $\{i_m \in [d] | m = 1, 2, \ldots, b\}$ uniformly at random without replacement
  $\mathbb{I}_h = [e_{i_1}, e_{i_2}, \cdots, e_{i_b}]$
  $\Gamma_h = \frac{1}{n}\mathbb{I}_h^T X X^T \mathbb{I}_h + \lambda \mathbb{I}_h^T \mathbb{I}_h$
  $\Delta w_h = \Gamma_h^{-1}\left(-\lambda \mathbb{I}_h^T w_{h-1} - \frac{1}{n}\mathbb{I}_h^T X z_{h-1} + \frac{1}{n}\mathbb{I}_h^T X y\right)$
  $w_h = w_{h-1} + \mathbb{I}_h \Delta w_h$
  $z_h = z_{h-1} + X^T \mathbb{I}_h \Delta w_h$
Output w_H

Communication
(every iteration)

Algorithm 2 Communication-Avoiding Block Coordinate Descent (CA-BCD) Algorithm

Input: $X \in \mathbb{R}^{d \times n}, y \in \mathbb{R}^n,\ H > 1,\ w_0 \in \mathbb{R}^d,\ b \in \mathbb{Z}_+$ s.t. $b \leq d$
for $k = 0, 1, \cdots, \frac{H}{s}$ **do**
  for $j = 1, 2, \cdots, s$ **do**
    choose $\{i_m \in [d] | m = 1, 2, \ldots, b\}$ uniformly at random without replacement
    $\mathbb{I}_{sk+j} = [e_{i_1}, e_{i_2}, \cdots, e_{i_b}]$
  let $Y = \left[\mathbb{I}_{sk+1}, \mathbb{I}_{sk+2}, \cdots, \mathbb{I}_{sk+s}\right]^T X$.
  compute the Gram matrix, $G = \frac{1}{n} Y Y^T + \lambda I$.
  for $j = 1, 2, \cdots, s$ **do**
    Γ_{sk+j} are the $b \times b$ diagonal blocks of G.
    $\Delta w_{sk+j} = \Gamma_{sk+j}^{-1}\Big(-\lambda \mathbb{I}_{sk+j}^T w_{sk} - \lambda \sum_{t=1}^{j-1}\left(\mathbb{I}_{sk+j}^T \mathbb{I}_{sk+t} \Delta w_{sk+t}\right) - \frac{1}{n}\mathbb{I}_{sk+j}^T X z_{sk}$
      $- \frac{1}{n}\sum_{t=1}^{j-1}\left(\mathbb{I}_{sk+j}^T X X^T \mathbb{I}_{sk+t} \Delta w_{sk+t}\right) + \frac{1}{n}\mathbb{I}_{sk+j}^T X y\Big)$
    $w_{sk+j} = w_{sk+j-1} + \mathbb{I}_{sk+j} \Delta w_{sk+j}$
    $z_{sk+j} = z_{sk+j-1} + X^T \mathbb{I}_{sk+j} \Delta w_{sk+j}$
Output w_H

Communication
(every outer iteration)

No communication

Theoretical Bounds

No free lunch: Reduce latency, but increase flops and bandwidth.

Suppose we perform **H iterations.**

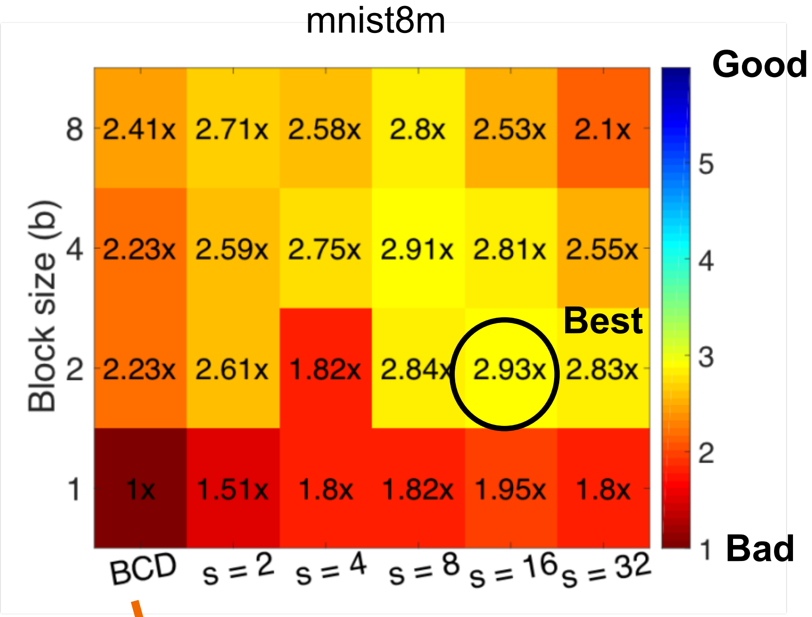
	Flops	Bandwidth	Latency
BCD	$O\left(\frac{Hb^2n}{P} + Hb^3\right)$	$O(Hb^2)$	$O(H \log P)$
CA-BCD	$O\left(\frac{Hsb^2n}{P} + Hb^3\right)$	$O(Hsb^2)$	$O\left(\frac{H}{s} \log P\right)$

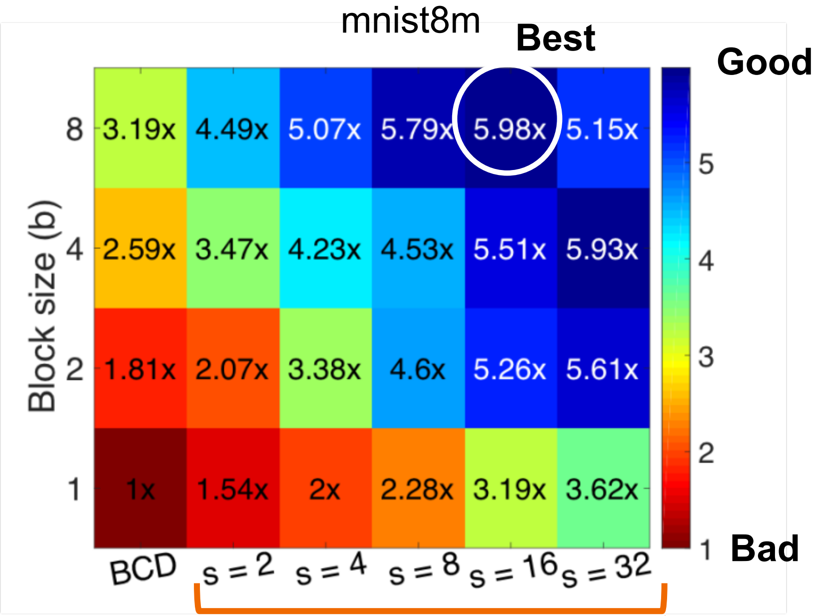
Similar bounds for **sparse.**

So, **Latency-** or **Synchronization-Avoiding.**

Speedups (Ridge Regression)

8M samples x 768 features





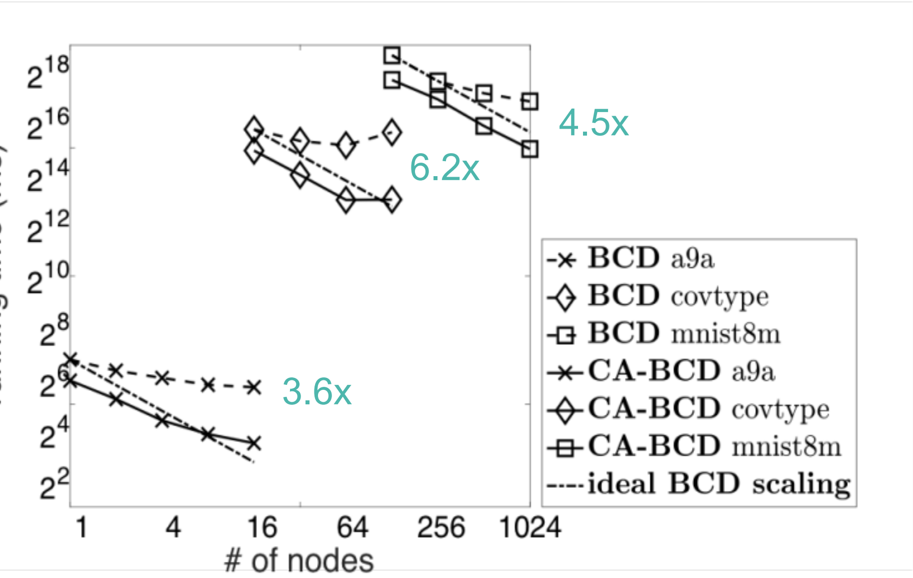
64 nodes of Edison

1K nodes of Edison

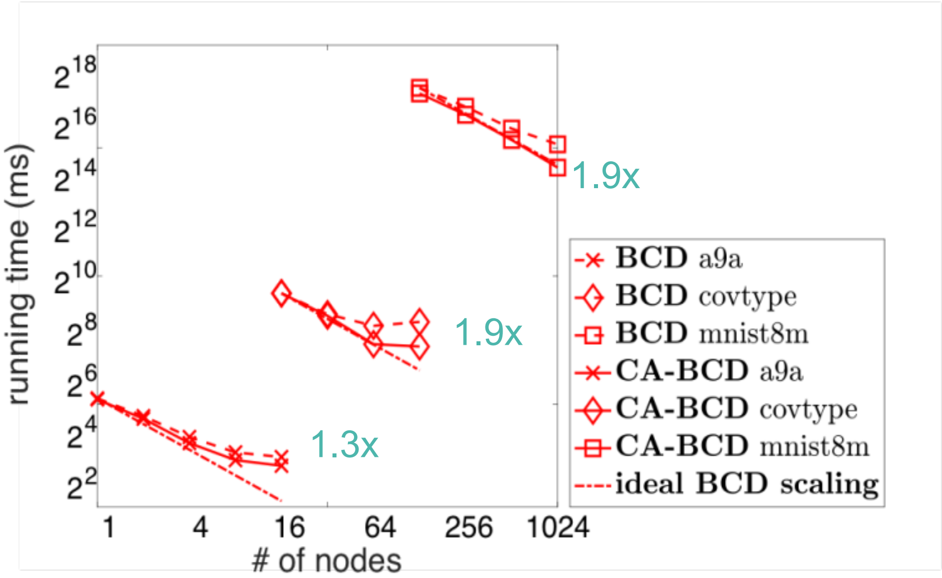
Standard Algorithm

Communication-Avoiding

Strong Scaling (Ridge Regression)



Blocksize = 1

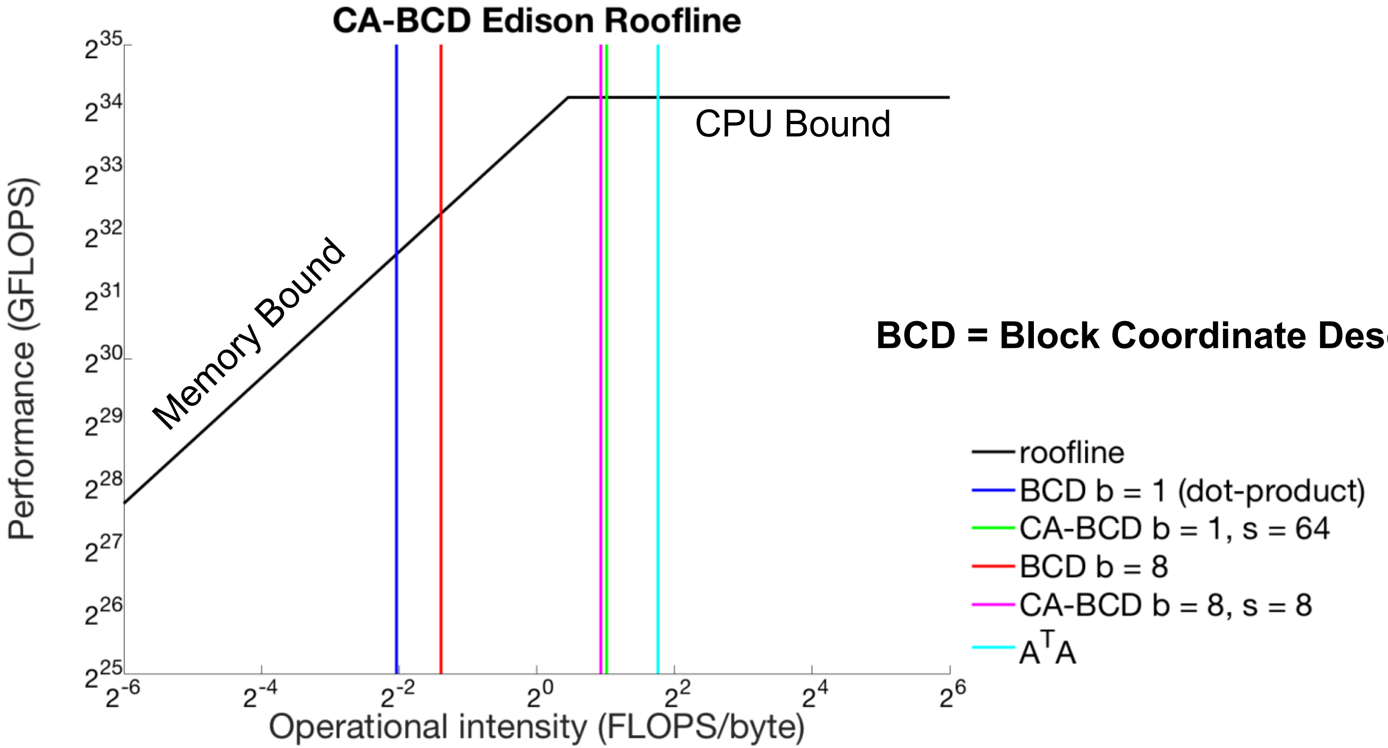


Blocksize = 8

Strong scaling = fixed problem size as P increases.

Theoretical Peak Attainable

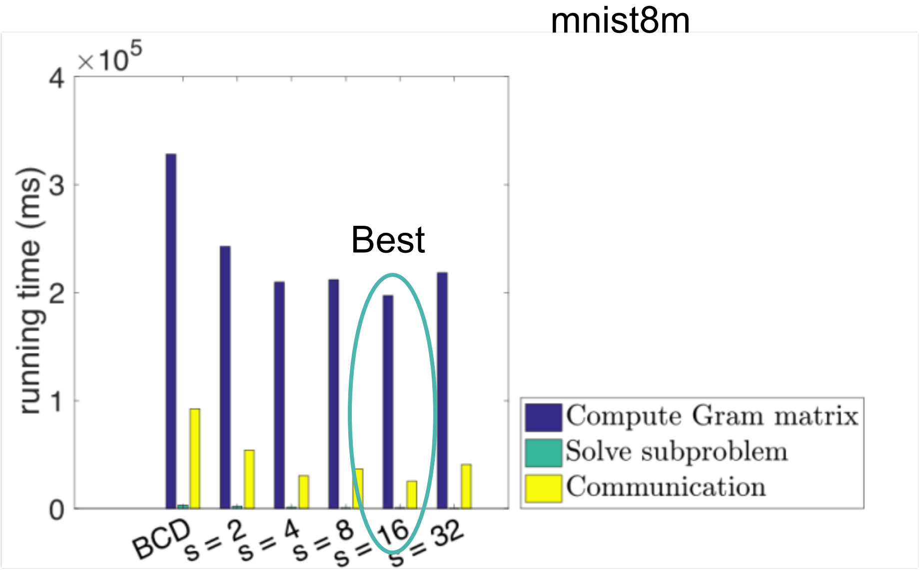
Intel “Ivy Bridge”: ~19 Gflops
64 GB DDR3 1866 MHz: 14GB/s



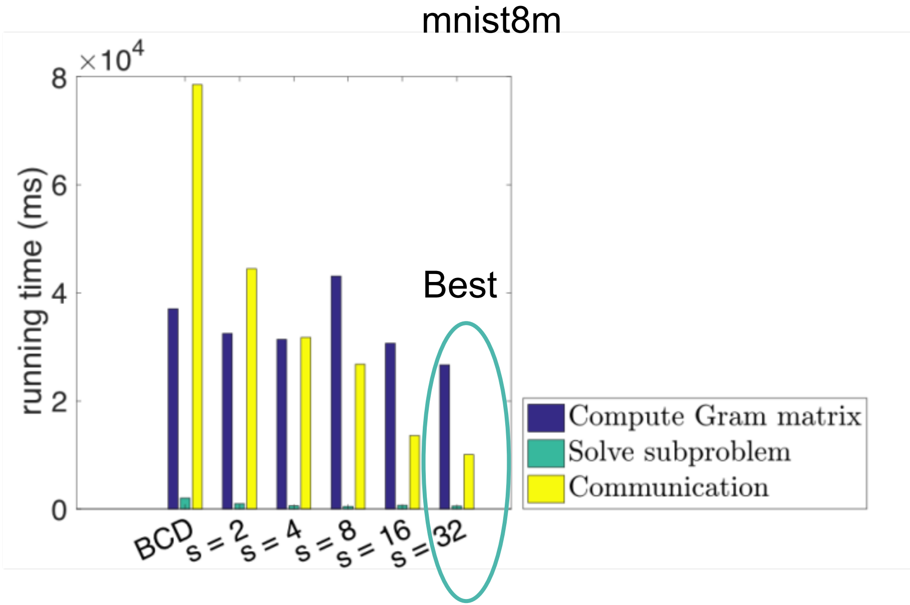
BCD = Block Coordinate Descent

<u>Dataset</u>
a9a: 32k x 123 matrix.
Sparsity: 11.28%

Running Time Breakdown (Ridge Regression)



64 nodes of Edison

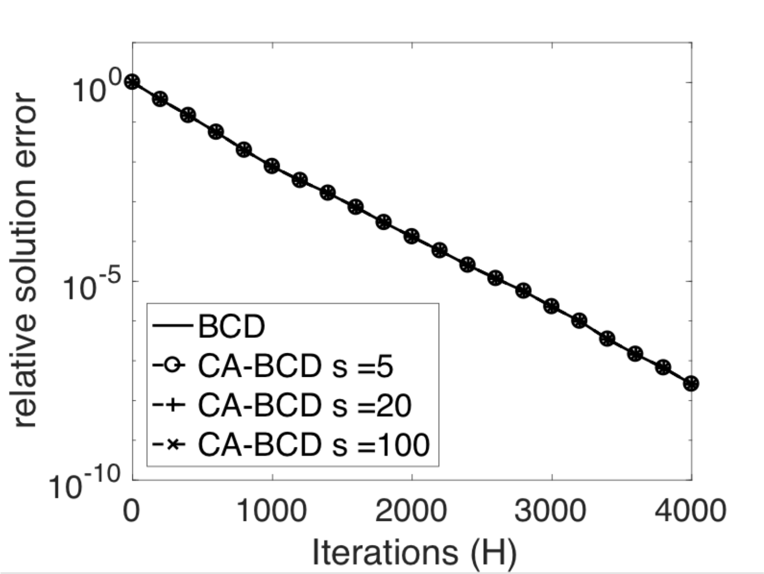


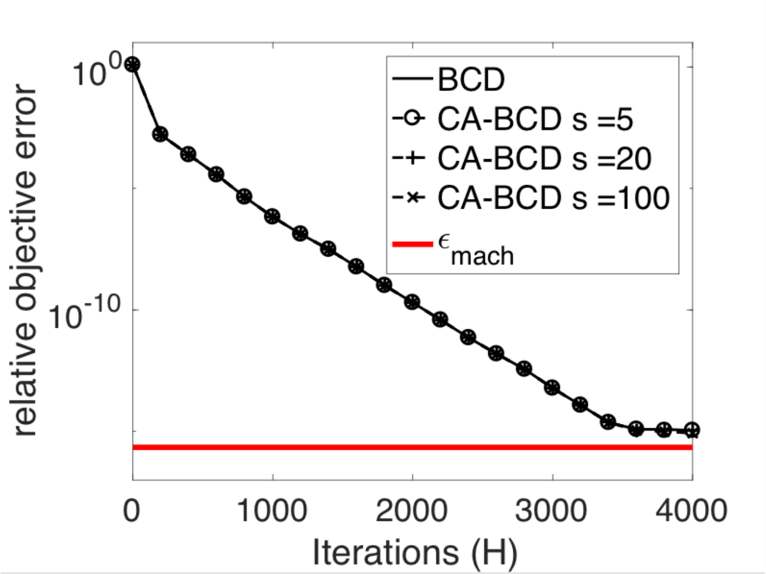
1K nodes of Edison

Blocksize = 1

Numerical Stability (Ridge Regression)

a9a dataset. 32k samples by 123 features. 11.28% non-zeros.





Blocksize = 16, $\boldsymbol{\sigma}_{min}$ = 4.884e-6
$\boldsymbol{\lambda}$ = 1000*$\boldsymbol{\sigma}_{min}$

Not Just Ridge Regression

Applies to **proximal methods** (e.g. LASSO, Group LASSO, elastic-net).

Support Vector Machines (binary classification).

And **associated Dual problems**.

Kernel Ridge and SVM (current work).

more generally to **Generalized Linear Models?** (future work).

Other optimization methods: **Stochastic Gradient Descent?** (future work).

Summary and Future Work

Large speedups when **latency dominates**.

Provably communication-avoiding.

CA-technique applies to **non-linear optimization**.

How far can we go (e.g. Logistic regression)?

Speedups on **other platforms and frameworks**?

Example: Cloud + Spark is latency dominated.

Expect greater speedups!

Problem	MPI Speedup
Ridge Regression	Up to **6.1x**
Proximal Least-Squares	Up to **5.1x**
SVM	**Similar expected**

Questions?

Thanks!

Backup Slides

Running Time Breakdown

Strong Scaling (Ridge Regression)

Numerical Stability

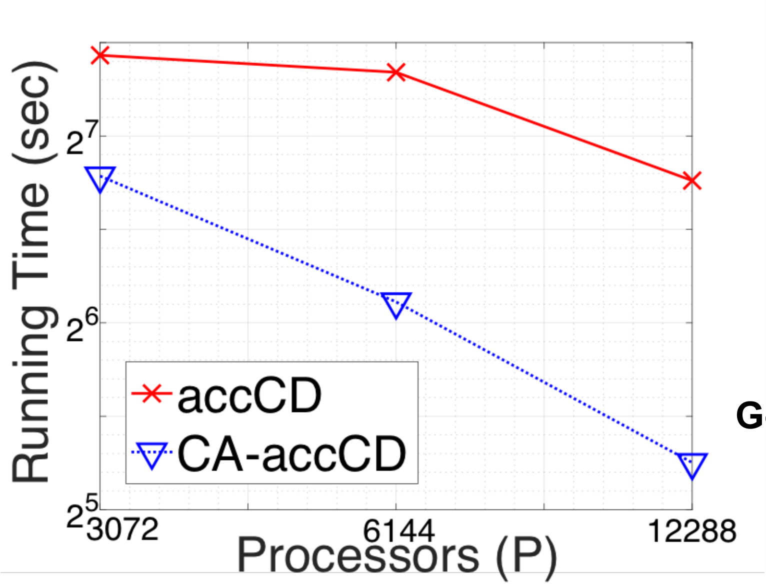
Proof of communication-avoidance (Ridge Regression)

CALASSOS: Scalable Proximal Methods

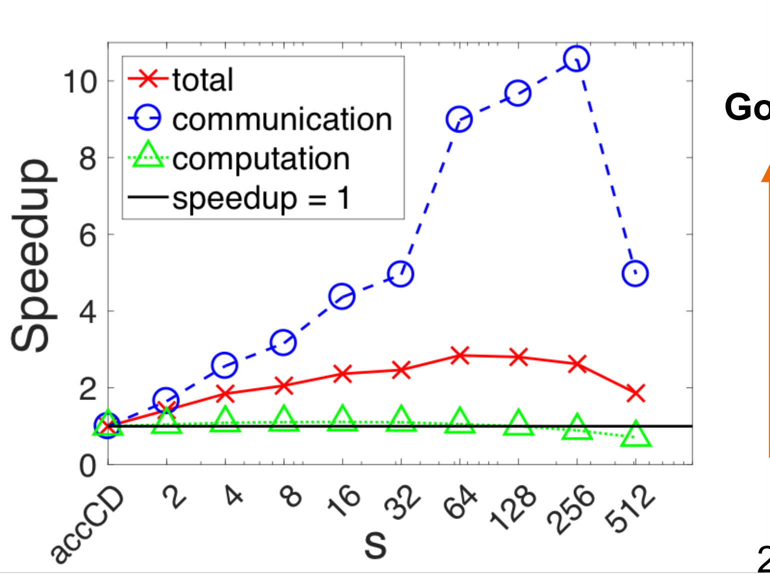
accCD = accelerated Coordinate Descent

CA-technique applies to accelerated methods.

Strong scaling and speedups on **Url dataset (2M by 3M).**



Good



Good

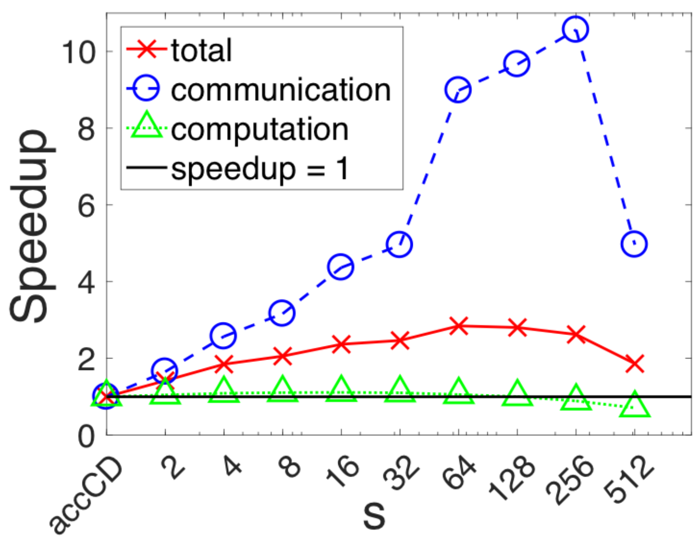
CALASSOS: Scalable Proximal Methods

CA-method perform **s^2 more flops.**

But still get **computation speedup.**

Due to BLAS-3 calls instead of BLAS-1 in non-CA.

BLAS-3 = Cache-efficient computation + higher flops rate.



Good

CA-SVM: Preliminary Results

Based on Dual Coordinate Descent for Linear SVM (Hsieh, et. al.)

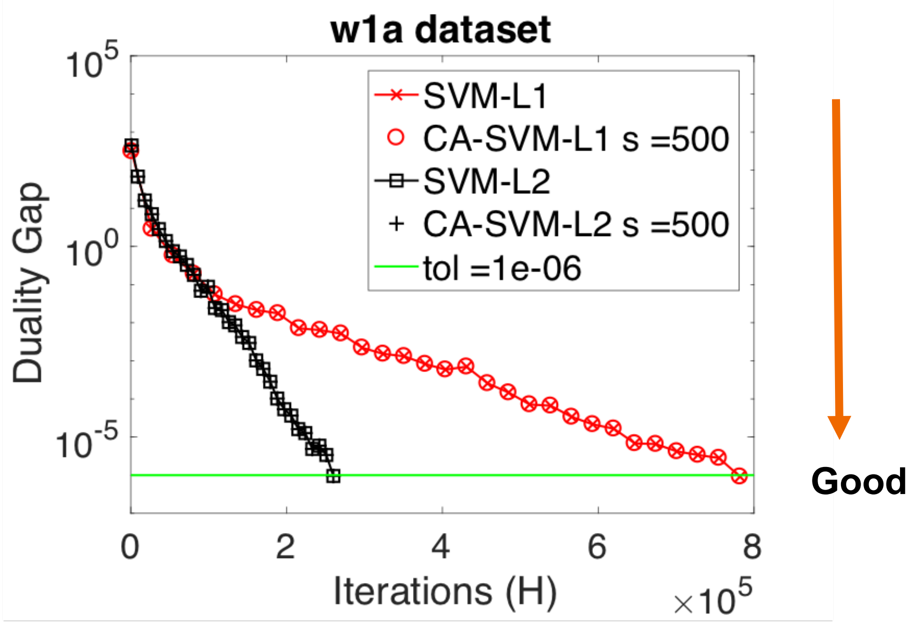
SVM-L1 = Hinge loss = *max(0, 1 –*$A^T_i x y_i$*)*

SVM-L2 = (Hinge loss)²

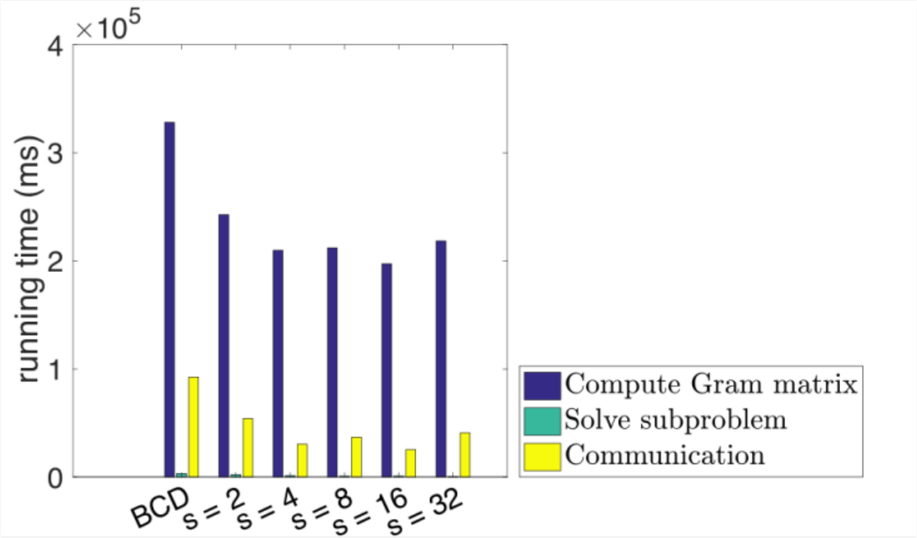
easier so, converges quickly.

Numerically stable **(unlike CA-Krylov)**!

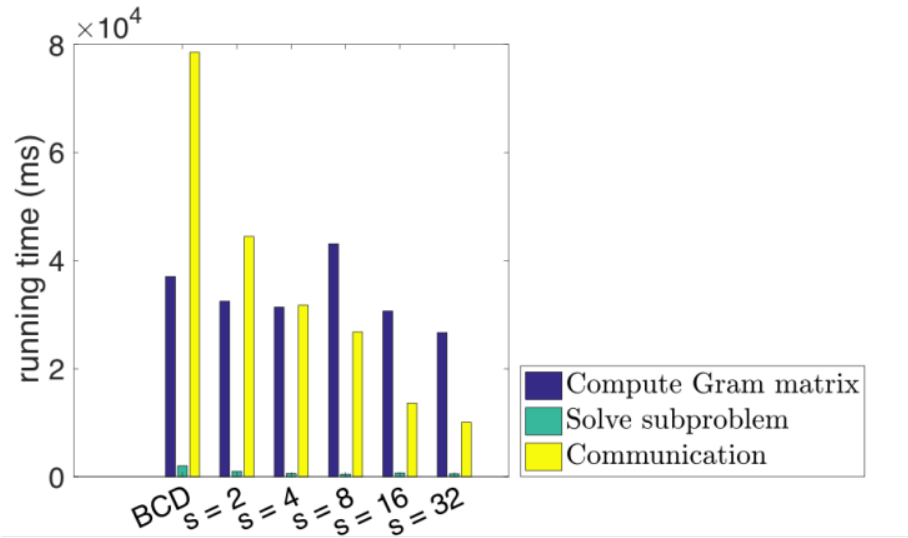
Ditto for Ridge and Proximal.



Running Time Breakdown (Ridge Regression)



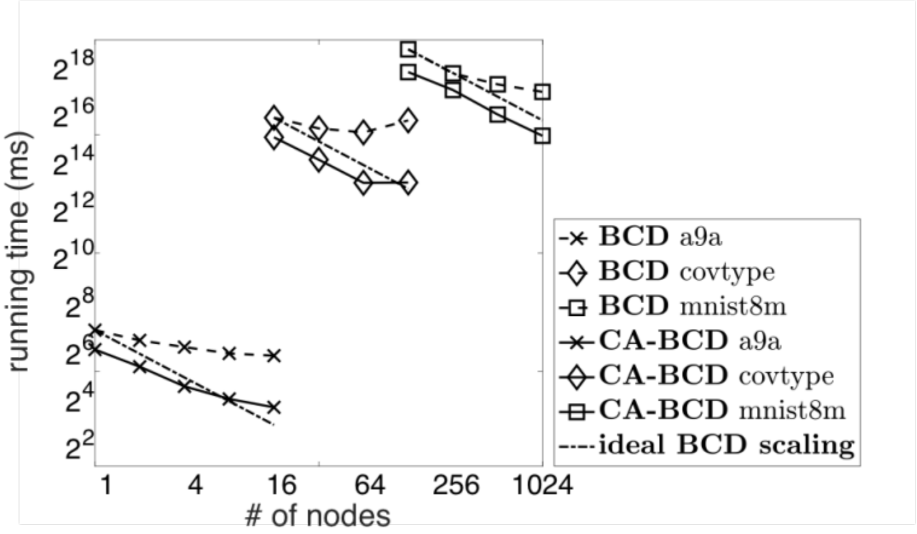
64 nodes of Edison



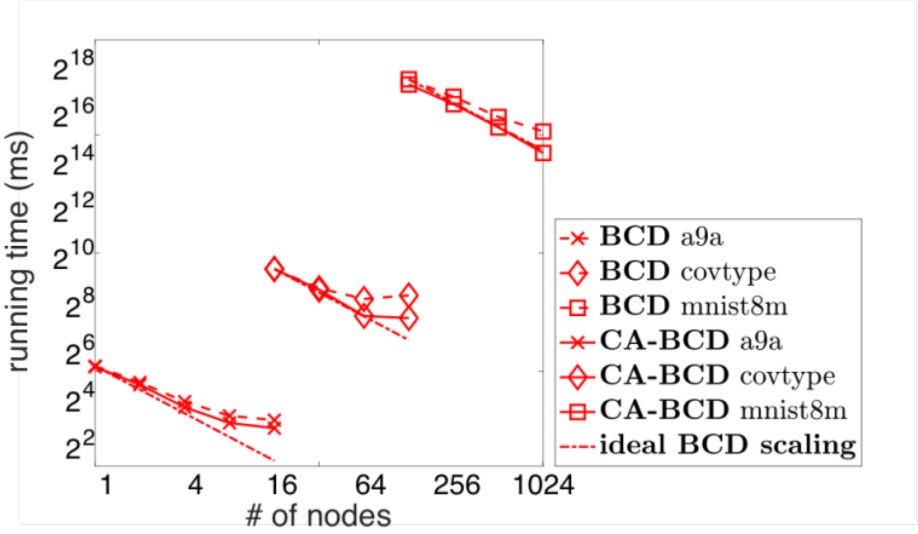
1K nodes of Edison

Blocksize = 1
(Coordinate Descent)

Strong Scaling (Ridge Regression)



Blocksize = 1



Blocksize = 8

Strong scaling = fixed problem size as **P** increases.

Provably Communication-Avoiding

No free lunch: Reduce latency, but increase flops and bandwidth.

Suppose we perform **H iterations.**

	Flops	Bandwidth	Latency
BCD	$O\left(\frac{Hb^2n}{P} + Hb^3\right)$	$O(Hb^2)$	$O(H \log P)$
CA-BCD	$O\left(\frac{H\mathbf{s}b^2n}{P} + Hb^3\right)$	$O(H\mathbf{s}b^2)$	$O\left(\frac{H}{\mathbf{s}} \log P\right)$

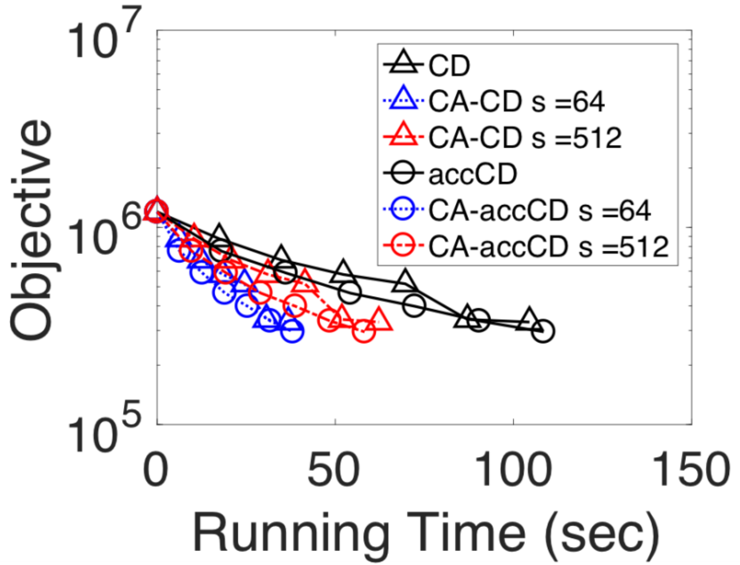
Similar bounds for **sparse.**

CALASSOS: Scalable Proximal Methods

Re-organized Accelerated BCD (Fercoq and Richtarik) for LASSO.

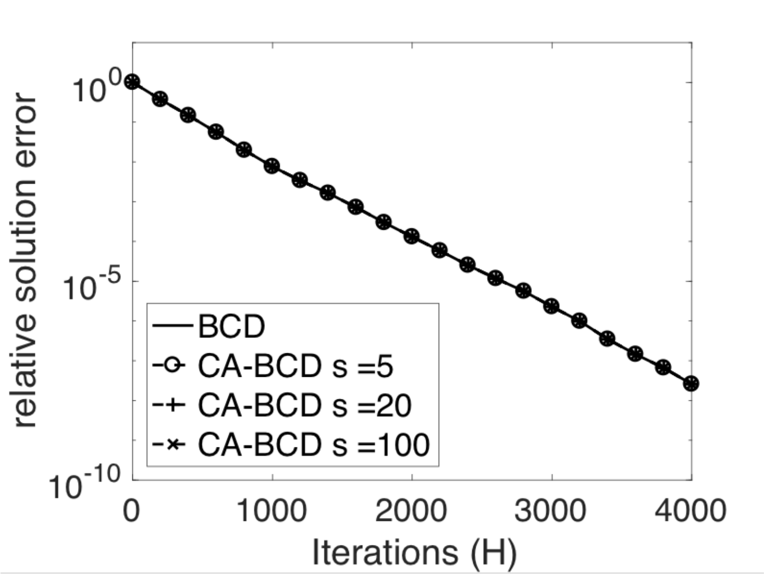
CA vs. non-CA on 1K nodes of Cray XC30.

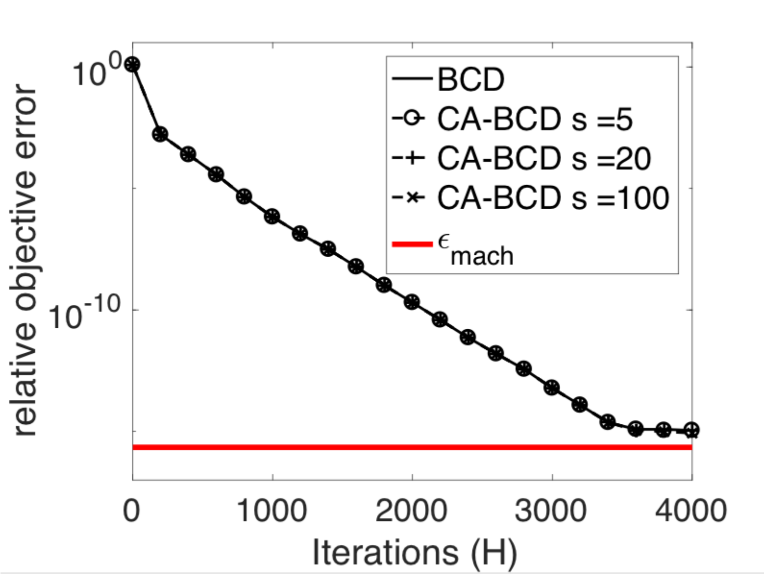
Choose **s carefully.**



Numerical Stability (Ridge Regression)

a9a dataset. 32k samples by 123 features. 11% non-zeros.

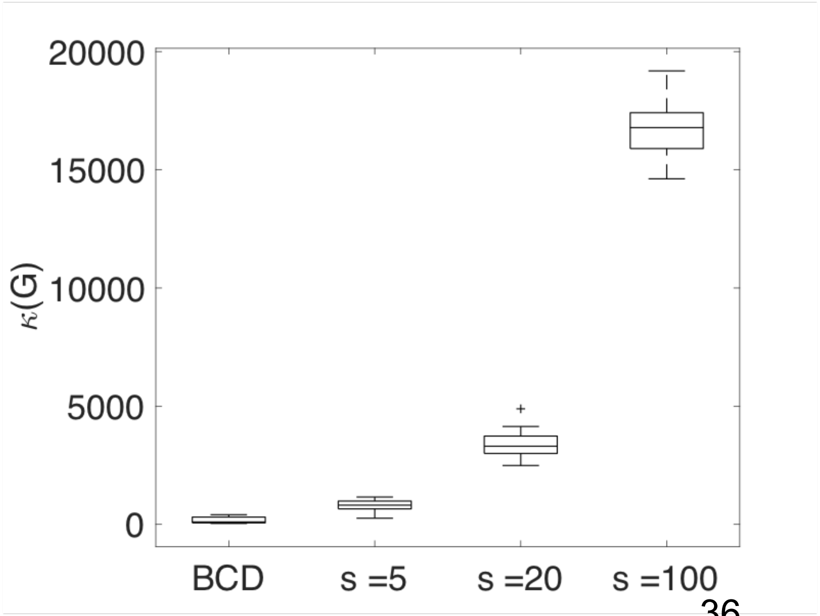




Numerical Stability (Ridge Regression)

We extract blocks from **G**.

So, **cond(G)** not a problem.



Numerical Stability (Ridge Regression)

CA-Krylov: numerical stability issues

Fix: Orthogonal polynomials and residual replacement strategies.

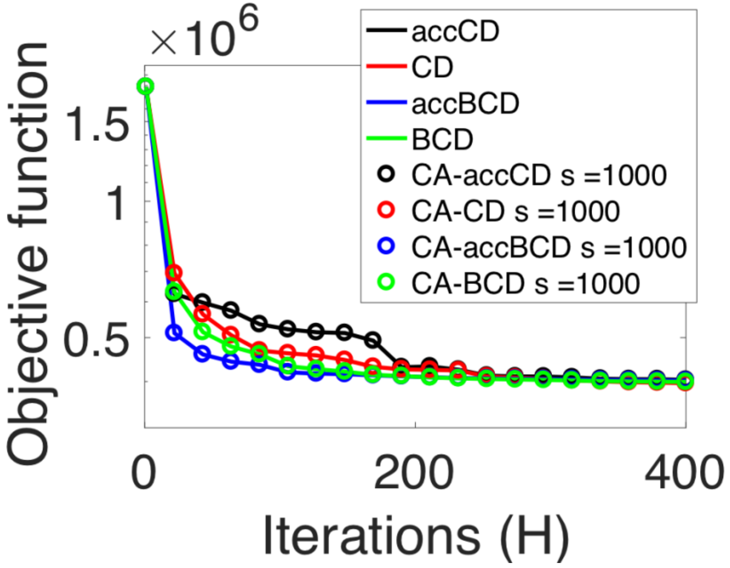
Dissertations by Hoemmen and Carson.

CA-Machine Learning: Magnitude of numerical error is small ~$O(10^{-15})$.

If we want **low-accuracy**, then CA-ML **essentially stable**.

CALASSOS: Scalable Proximal Methods

Re-organized Accelerated BCD (Fercoq and Richtarik) for LASSO.



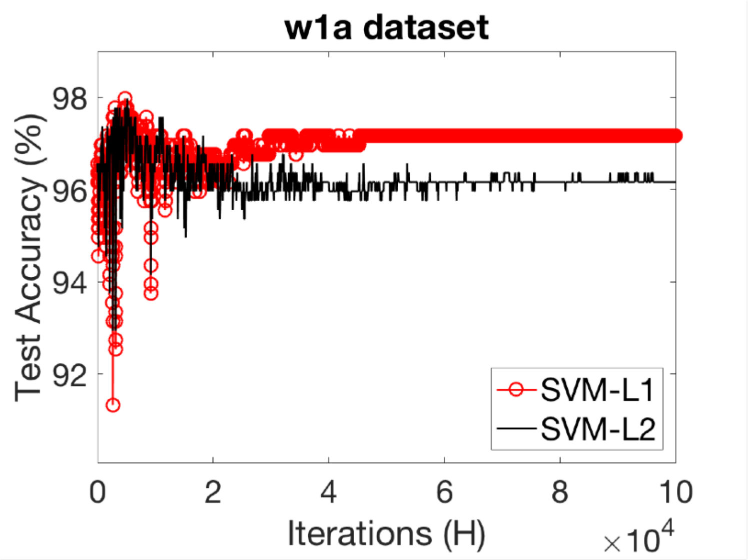
CA-SVM: Preliminary Results

Dual Coordinate Descent for Linear SVM (Hsieh, et. al.)

Once again, non-linearity in “inner loop”.

CA technique applies.

Similar speedups to LASSO expected.



Modeling Communication

Hardware Parameters

Algorithm Parameters

Running Time = Computation time + Communication time

(time per flop) **x** (# of flops)

(time per word) **x** (# of words) **+** (time per message) **x** (# of messages)