MS4: Minimizing Communication in Numerical Algorithms
Part I of II

Organizers: Oded Schwartz (*Hebrew University of Jerusalem*) and
Erin Carson (*New York University*)

Talks:

1. Communication-Avoiding Krylov Subspace Methods in Theory and Practice (***Erin Carson***)
2. Enlarged GMRES for Reducing Communication When Solving Reservoir Simulation Problems (***Hussam Al Daas***, Laura Grigori, Pascal Henon, Philippe Ricoux)
3. CA-SVM: Communication-Avoiding Support Vector Machines on Distributed Systems (***Yang You***)
4. Outline of a New Roadmap to Permissive Communication and Applications That Can Benefit (***James A. Edwards*** and Uzi Vishkin)

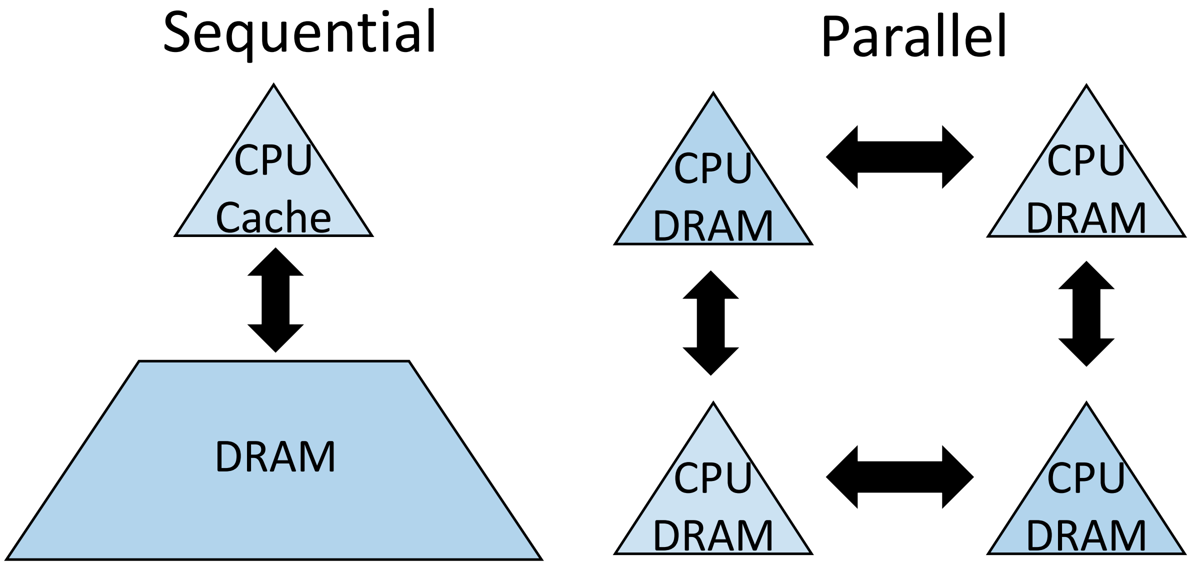
Communication-Avoiding Krylov Subspace Methods in Theory and Practice

Erin Carson
Courant Institute @ NYU

April 12, 2016
SIAM PP 16

What is communication?

- Algorithms have two costs: **computation** and **communication**
  - **Communication : moving data** between levels of memory hierarchy (sequential), between processors (parallel)



- On today's computers, communication is expensive, computation is cheap, in terms of both time and energy!

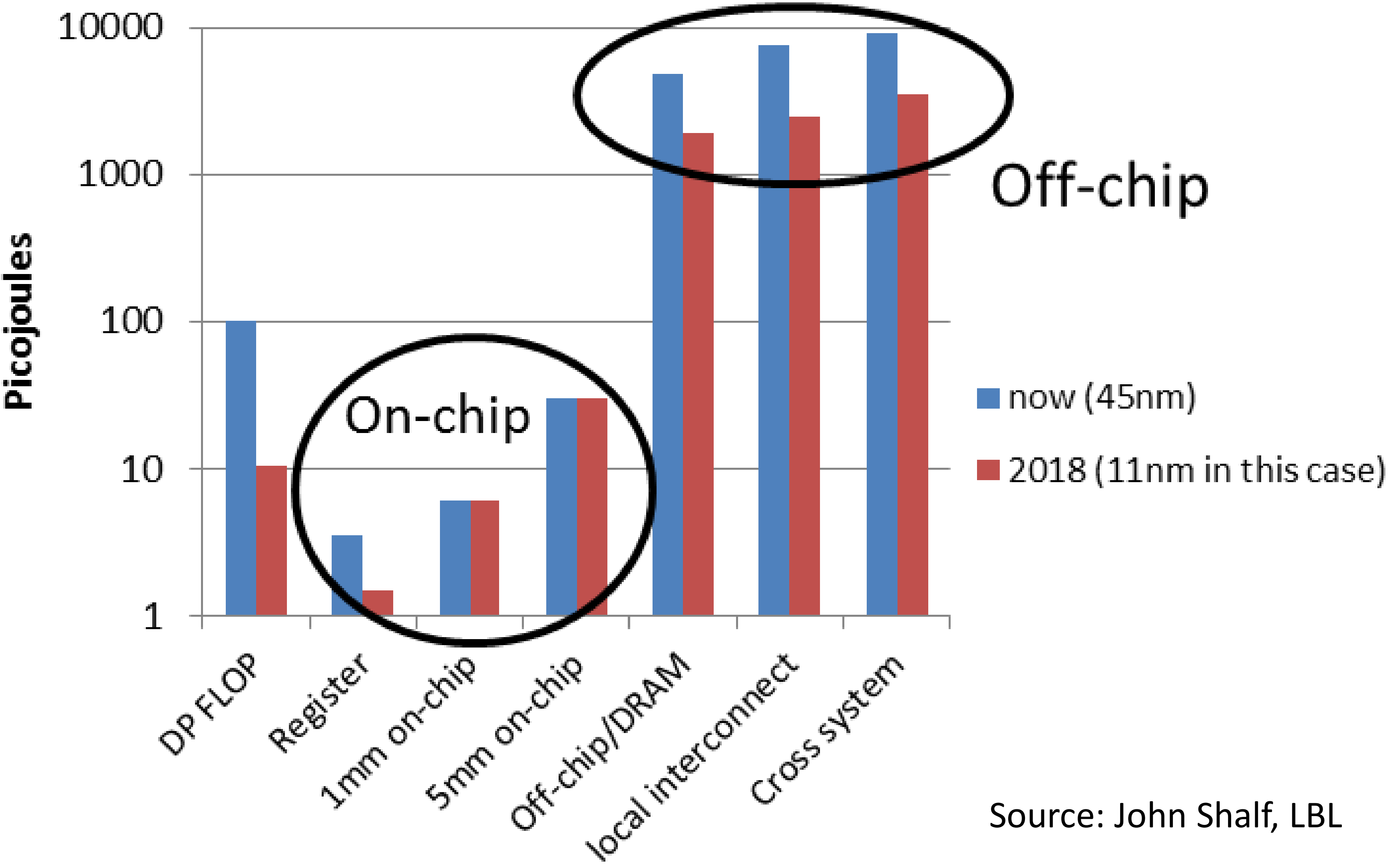
Future exascale systems

	Petascale Systems (2009)
System Peak	$2 \cdot 10^{15}$ flops/s
Node Memory Bandwidth	25 GB/s
Total Node Interconnect Bandwidth	3.5 GB/s
Memory Latency	100 ns
Interconnect Latency	1 μs

*Sources: from P. Beckman (ANL), J. Shalf (LBL), and D. Unat (LBL)

- Gaps between communication/computation cost only growing larger in future systems
- **Avoiding communication will be essential for applications at exascale!**

Minimize communication to save energy



Source: John Shalf, LBL

Work in CA algorithms

- For both dense and sparse linear algebra…
  - Prove lower bounds on communication cost of an algorithm
  - Design new algorithms and implementations that meet these those bounds

- More recently, extending communication-avoiding ideas to Machine Learning and optimization domains

Lots of speedups…

- Up to **12x** faster for 2.5D matmul on 64K core IBM BG/P
- Up to **3x** faster for tensor contractions on 2K core Cray XE/6
- Up to **6.2x** faster for All-Pairs-Shortest-Path on 24K core Cray CE6
- Up to **2.1x** faster for 2.5D LU on 64K core IBM BG/P
- Up to **11.8x** faster for direct N-body on 32K core IBM BG/P
- Up to **13x** faster for Tall Skinny QR on Tesla C2050 Fermi NVIDIA GPU
- Up to **6.7x** faster for symeig(band A) on 10 core Intel Westmere
- Up to **2x** faster for 2.5D Strassen on 38K core Cray XT4
- Up to **4.2x** faster for MiniGMG benchmark bottom solver, using CA-BICGSTAB (**2.5x** for overall solve), **2.5x** / **1.5x** for combustion simulation code
- Up to **42x** for Parallel Direct 3-Body

These and many more recent papers available at bebop.cs.berkeley.edu

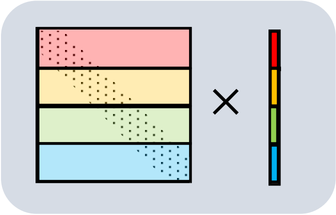
Krylov solvers: limited by communication

In terms of linear algebra operations:

“Add a dimension to $\mathcal{K}_m$”

→ Sparse Matrix-Vector Multiplication (SpMV)

- Parallel: comm. vector entries w/ neighbors
- Sequential: read A/vectors from slow memory

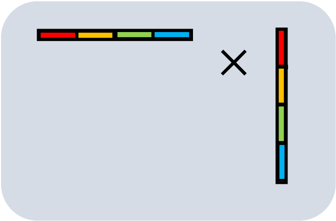


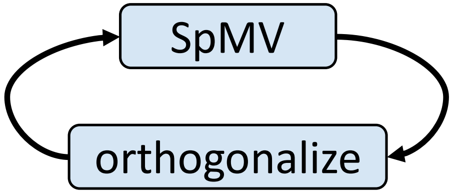
“Orthogonalize (with respect to some $\mathcal{L}_m$)”

→ Inner products

Parallel: global reduction (All-Reduce)

Sequential: multiple reads/writes to slow memory





Dependencies between communication-bound kernels in each iteration limit performance!

The classical Lanczos method

```
Given: initial vector $v_1$ with $||v_1||_2 = 1$
$u_1 = Av_1$
**for** $i = 1, 2, \dots,$ until convergence **do**
        $\alpha_i = \boxed{v_i^T u_i}$
        $w_i = u_i - \alpha_i v_i$
        $\beta_{i+1} = \boxed{||w_i||_2}$
        $v_{i+1} = w_i / \beta_{i+1}$
        $u_{i+1} = \boxed{Av_{i+1}} - \beta_{i+1} v_i$
**end for**
```

Inner products

SpMV

Communication-Avoiding KSMs

- Idea: Compute blocks of s iterations at once
  - Communicate every s iterations instead of every iteration
  - **Reduces communication cost by $\boldsymbol{O(s)}$!**
    - (latency in parallel, latency and bandwidth in sequential)

- An idea rediscovered many times…
- First related work: s-dimensional steepest descent, least squares
  - Khabaza (‘63), Forsythe (‘68), Marchuk and Kuznecov (‘68)
- Flurry of work on s-step Krylov methods in ‘80s/early ‘90s: see, e.g., Van Rosendale, 1983; Chronopoulos and Gear, 1989
  - Goals: increasing parallelism, avoiding I/O, increasing “convergence rate”

- Resurgence of interest in recent years due to growing problem sizes; growing relative cost of communication

Communication-Avoiding KSMs: CA-Lanczos

- Main idea: Unroll iteration loop by a factor of s; split iteration loop into an outer loop (k) and an inner loop (j)
- Key observation: starting at some iteration $i \equiv sk + j$,

$$v_{sk+j},\ u_{sk+j} \in \mathcal{K}_{s+1}(A, v_{sk+1}) + \mathcal{K}_{s+1}(A, u_{sk+1}) \quad \text{for} \quad j \in \{1, \dots, s+1\}$$

For each block of s steps:

- Compute “basis matrix”: $\mathcal{Y}_k$ such that

$$\mathrm{span}(\mathcal{Y}_k) = \mathcal{K}_{s+1}(A, v_{sk+1}) + \mathcal{K}_{s+1}(A, u_{sk+1})$$

  - $O(s)$ SpMVs, requires reading A/communicating vectors only once using “matrix powers kernel”
- Orthogonalize: $\mathcal{G}_k = \mathcal{Y}_k^T \mathcal{Y}_k$
  - One global reduction
- Perform s iterations of updates for n-vectors by updating their $O(s)$ coordinates in $\mathcal{Y}_k$
  - No communication

The CA-Lanczos method

```
Given: initial vector $v_1$ with $||v_1||_2 = 1$
$u_1 = Av_1$
for $k = 0, 1, \dots,$ until convergence do
    Compute $\mathcal{Y}_k$, compute $\mathcal{G}_k = \mathcal{Y}_k^T \mathcal{Y}_k$
    Let $v'_{k,1} = e_1,\ u'_{k,1} = e_{s+2}$
    for $j = 1, \dots, s$ do
        $\alpha_{sk+j} = v'^T_{k,j} \mathcal{G}_k u'_{k,j}$
        $w'_{k,j} = u'_{k,j} - \alpha_{sk+j} v'_{k,j}$
        $\beta_{sk+j+1} = \left(w'^T_{k,j} \mathcal{G}_k w'_{k,j}\right)^{1/2}$
        $v'_{k,j+1} = w'_{k,j} / \beta_{sk+j+1}$
        $u'_{k,j+1} = \mathcal{B}_k v'_{k,j+1} - \beta_{sk+j+1} v'_{k,j}$
    end for
    Compute $v_{sk+s+1} = \mathcal{Y}_k v'_{k,s+1},\ u_{sk+s+1} = \mathcal{Y}_k u'_{k,s+1}$
end for
```

via CA Matrix Powers Kernel

Global reduction to compute $\mathcal{G}_k$

Local computations: no communication!

Complexity comparison

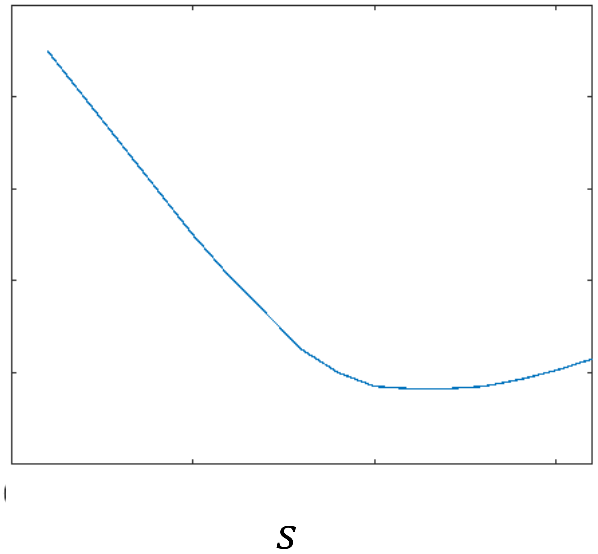
Example of parallel (per processor) complexity for s iterations of Classical Lanczos vs. CA-Lanczos for a 2D 9-point stencil:

(Assuming each of p processors owns n/p rows of the matrix and $s \leq \sqrt{n/p}$)

	Flops		Words Moved		Messages	
	SpMV	Orth.	SpMV	Orth.	SpMV	Orth.
Classical CG	$\frac{sn}{p}$	$\frac{sn}{p}$	$s\sqrt{n/p}$	$s \log_2 p$	s	$s \log_2 p$
CA-CG	$\frac{sn}{p}$	$\frac{s^2 n}{p}$	$s\sqrt{n/p}$	$s^2 \log_2 p$	1	$\log_2 p$

All values in the table meant in the Big-O sense (i.e., lower order terms and constants not included)

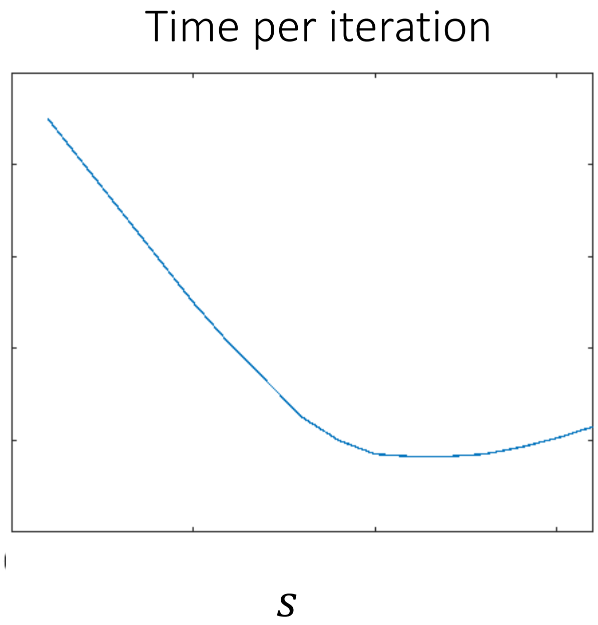
Time per iteration



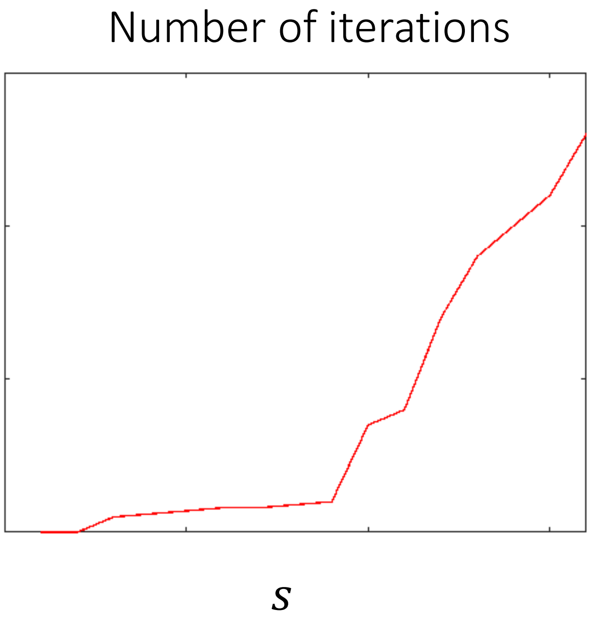
From theory to practice

- CA-KSMs are mathematically equivalent to classical KSMs
- But can behave much differently in finite precision!
- Roundoff error bounds generally grow with increasing s
- Two effects of roundoff error:

1. **Decrease in accuracy** → Tradeoff: increasing blocking factor s past a certain point: accuracy limited
2. **Delay of convergence** → Tradeoff: increasing blocking factor s past a certain point: no speedup expected



Time per iteration



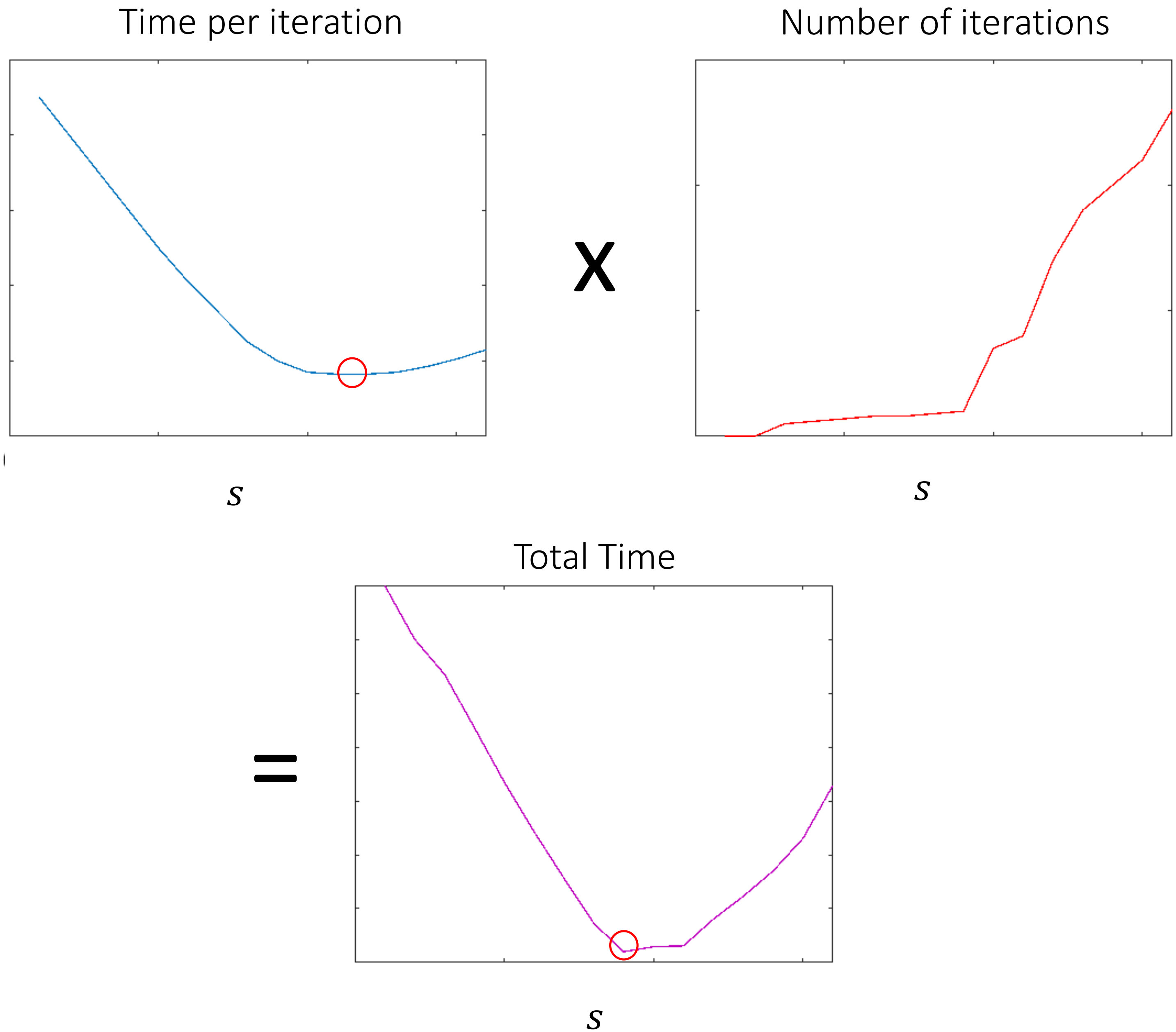
Number of iterations

Optimizing iterative method runtime

- Want to minimize total time of iterative solver

Runtime = (time/iteration) x (# iterations)

- Time per iteration determined by matrix/preconditioner structure, machine parameters, basis size, etc.
- Number of iterations depends on numerical properties of the matrix/preconditioner, basis size


Time per iteration
s
X
Number of iterations
s
=
Total Time
s

Optimizing iterative method runtime

- Want to minimize total time of iterative solver

Runtime = (time/iteration) x (# iterations)

- Speed per iteration determined by matrix/preconditioner structure, machine parameters, basis size, etc.
- Number of iterations depends on numerical properties of the matrix/preconditioner, basis size

- Traditional auto-tuners tune kernels (e.g., SpMV and QR) to optimize speed per iteration
- This misses a big part of the equation!
- Goal: Combine offline auto-tuning with online techniques for achieving desired accuracy and a good convergence rate
- Requires a **better understanding of behavior of iterative methods in finite precision**

Main results

- Bounds on accuracy and convergence rate for CA methods can be written in terms of those for classical methods times an amplification factor
  - Amplification factor depends on condition number of the s-step bases $\mathcal{Y}_k$ computed in each outer iteration
- These bounds can be used to design techniques to improve accuracy and convergence rate while still avoiding communication

Attainable accuracy of (CA)-CG

- Results for CG (Greenbaum, van der Vorst and Ye, others):

$$\|b - Ax_m - r_m\| \leq \varepsilon N^* \sum_{i=0}^{m} (1 + 2N)\|A\|\|\hat{x}_i\| + \|\hat{r}_i\|$$

- Results for CA-CG:

$$\|b - Ax_m - r_m\| \leq \varepsilon \boldsymbol{\Gamma_k} N^* \sum_{i=0}^{m} (1 + 2N)\|A\|\|\hat{x}_i\| + \|\hat{r}_i\|$$

where $\Gamma_k = \max_{\ell \leq k} \|\mathcal{Y}_\ell^+\|_2 \cdot \||\mathcal{Y}_\ell|\|_2$

- Bound can be used for designing a “Residual Replacement” strategy for CA-CG (based on van der Vorst and Ye, 1999)
  - In tests, CA-CG accuracy improved up to 7 orders of magnitude for little additional cost

Convergence and accuracy of CA-Lanczos

- Chris Paige's results for classical Lanczos:

  loss of orthogonality → eigenvalue convergence

  if $\varepsilon n \leq \frac{1}{12}$

- This (and other results of Paige) also hold for CA-Lanczos if:

  $2\varepsilon(n+11s+15)\,\Gamma^2 \leq \frac{1}{12}$, where $\Gamma = \max_{\ell \leq k} \|\mathcal{Y}_\ell^+\|_2 \cdot \||\mathcal{Y}_\ell|\|_2$

  - i.e., $\max_{\ell \leq k} \|\mathcal{Y}_\ell^+\|_2 \cdot \||\mathcal{Y}_\ell|\|_2 \leq \left(24\epsilon(n+11s+15)\right)^{-1/2}$

- We could approximate this constraint:

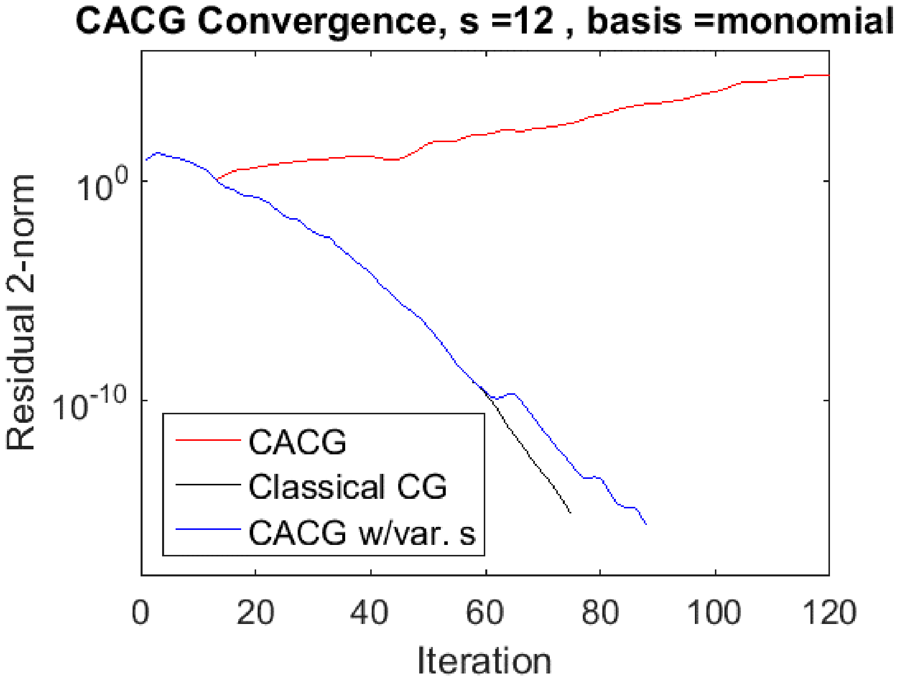
$$\kappa(\mathcal{Y}_k) \leq 1/\sqrt{\epsilon n}$$

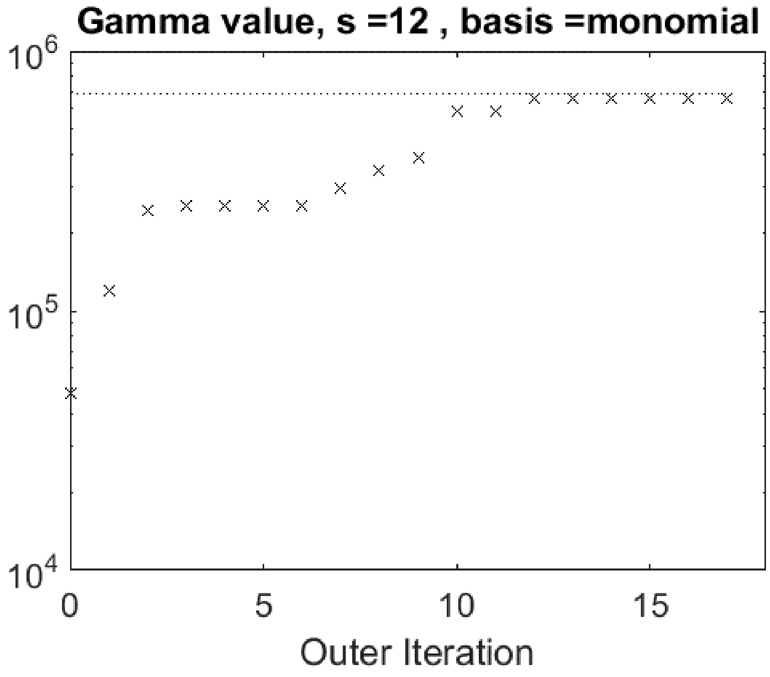
and use it to design a better algorithm!

Dynamic basis size

- Auto-tune to find best s based on machine, sparsity structure; use this as $s_{\max}$
- In each outer iteration, select largest $s \leq s_{\max}$ such that $\kappa(\mathcal{Y}_k) \leq 1/\sqrt{\epsilon n}$
- Benefit: Maintain acceptable convergence rate regardless of user's choice of s
- Cost: Incremental condition number estimation in each outer iteration; potentially wasted SpMVs in each outer iteration

s values used = (6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 4, 4, 5, 5)





MS4: Minimizing Communication in Numerical Algorithms
Part I of II

Talks:

1. Communication-Avoiding Krylov Subspace Methods in Theory and Practice (***Erin Carson***)
2. Enlarged GMRES for Reducing Communication When Solving Reservoir Simulation Problems (***Hussam Al Daas***, Laura Grigori, Pascal Henon, Philippe Ricoux)
3. CA-SVM: Communication-Avoiding Support Vector Machines on Distributed Systems (***Yang You***)
4. Outline of a New Roadmap to Permissive Communication and Applications That Can Benefit (***James A. Edwards*** and Uzi Vishkin)

MS12: Minimizing Communication in Numerical Algorithms
Part II of II

3:20 PM - 5:00 PM, *Room: Salle des theses*

1. Communication-Efficient Evaluation of Matrix Polynomials (***Sivan A. Toledo***)
2. Communication-Optimal Loop Nests (***Nicholas Knight***)
3. Write-Avoiding Algorithms (***Harsha Vardhan Simhadri***)
4. Lower Bound Techniques for Communication in the Memory Hierarchy (***Gianfranco Bilardi***)

Thank you!

Email: erinc@cims.nyu.edu

Website: http://cims.nyu.edu/~erinc/

Matlab code: https://github.com/eccarson/ca-ksms

Residual Replacement Strategy

- Improve accuracy by **replacing updated residual $\boldsymbol{r_m}$ by the true residual $\boldsymbol{b - Ax_m}$** in certain iterations
  - Related work for classical CG: van der Vorst and Ye (1999)

- Choose when to replace r_m with $b - Ax_m$ to meet two constraints:
  1. $\|b - Ax_m - r_m\|$ is small
  2. Convergence rate is maintained (avoid large perturbations to finite precision CG recurrence)

- Based on derived bound on deviation of residuals, can devise a residual replacement strategy for CA-CG and CA-BICG

- Implementation has **negligible cost** → residual replacement strategy allows **both speed and accuracy!**

Residual Replacement for CA-CG

- Use computable bound for $\|b - Ax_{sk+j} - r_{sk+j}\|$ to update d_{sk+j}, an estimate of error in computing r_{sk+j}, in each iteration
- Set threshold $\hat{\varepsilon} \approx \sqrt{\varepsilon}$, replace whenever $\boxed{d_{sk+j}}/\|r_{sk+j}\|$ reaches threshold

Pseudo-code for residual replacement with group update for CA-CG:

```
if  $d_{sk+j-1} \le \hat{\varepsilon}\|r_{sk+j-1}\|$ and $d_{sk+j} > \hat{\varepsilon}\|r_{sk+j}\|$ and $d_{sk+j} > 1.1 d_{init}$
        $z = z + Y_k x'_{k,j} + x_{sk}$
        $x_{sk+j} = 0$
        $r_{sk+j} = b - Az$
        $d_{init} = d_{sk+j} = \varepsilon\left((1 + 2N')\|A\|\|z\| + \|r_{sk+j}\|\right)$
        $p_{sk+j} = Y_k p'_{k,j}$
        break from inner loop and begin new outer loop
end
```

group update of approximate solution

set updated residual to true residual

A Computable Bound for CA-CG

- In each iteration, update error estimate d_{sk+j} by:

$O(s^3)$ flops per s iterations; ≤ 1 reduction per s iterations to compute $\left(|\hat{Y}_k|^T|\hat{Y}_k|\right)$

$O(s^2)$ flops per iteration; no communication

Estimated only once

Extra computation all lower order terms; communication only increased by *at most* factor of 2!

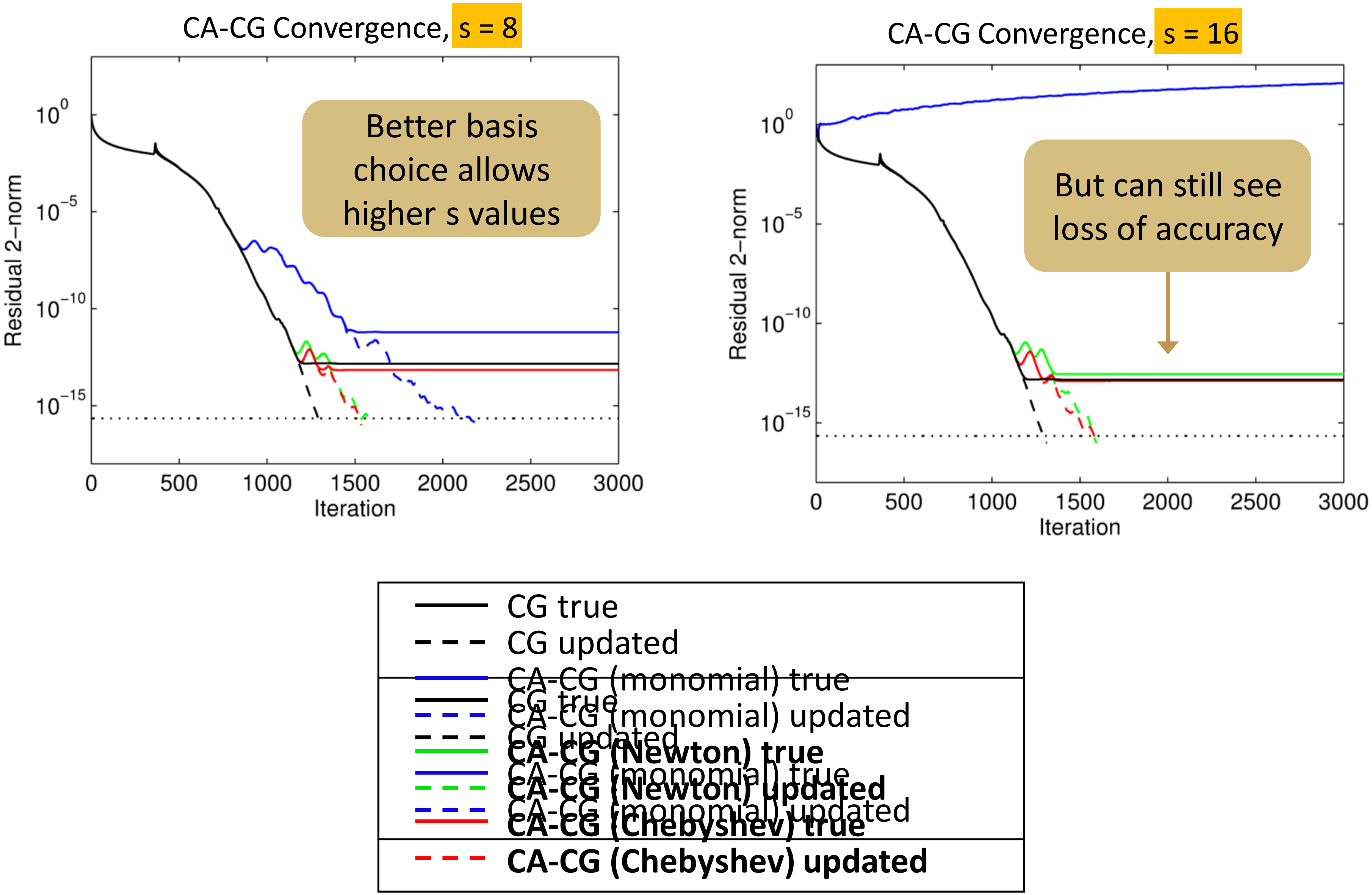
$$d_{sk+j} \equiv d_{sk+j-1}$$

$$+\varepsilon\left[(4+N')\left(\|A\|\,\left\||\hat{Y}_k|\cdot|\hat{x}'_{k,j}|\right\| + \left\||\hat{Y}_k|\cdot|B_k|\cdot|\hat{x}'_{k,j}|\right\|\right) + \left\||\hat{Y}_k|\cdot|\hat{r}'_{k,j}|\right\|\right]$$

$$+\varepsilon\begin{cases}\|A\|\|\hat{x}_{sk+s}\|+(2+2N')\|A\|\left\||\hat{Y}_k|\cdot|\hat{x}'_{k,s}|\right\|+N'\left\||\hat{Y}_k|\cdot|\hat{r}'_{k,s}|\right\|, & j=s\\ 0, & \text{otherwise}\end{cases}$$

where $N' = \max(N, 2s+1)$.

Model Problem: 2D Poisson, n = 262K, nnz = 1.3M, cond(A) $\approx 10^4$



CA-CG Convergence, s = 8

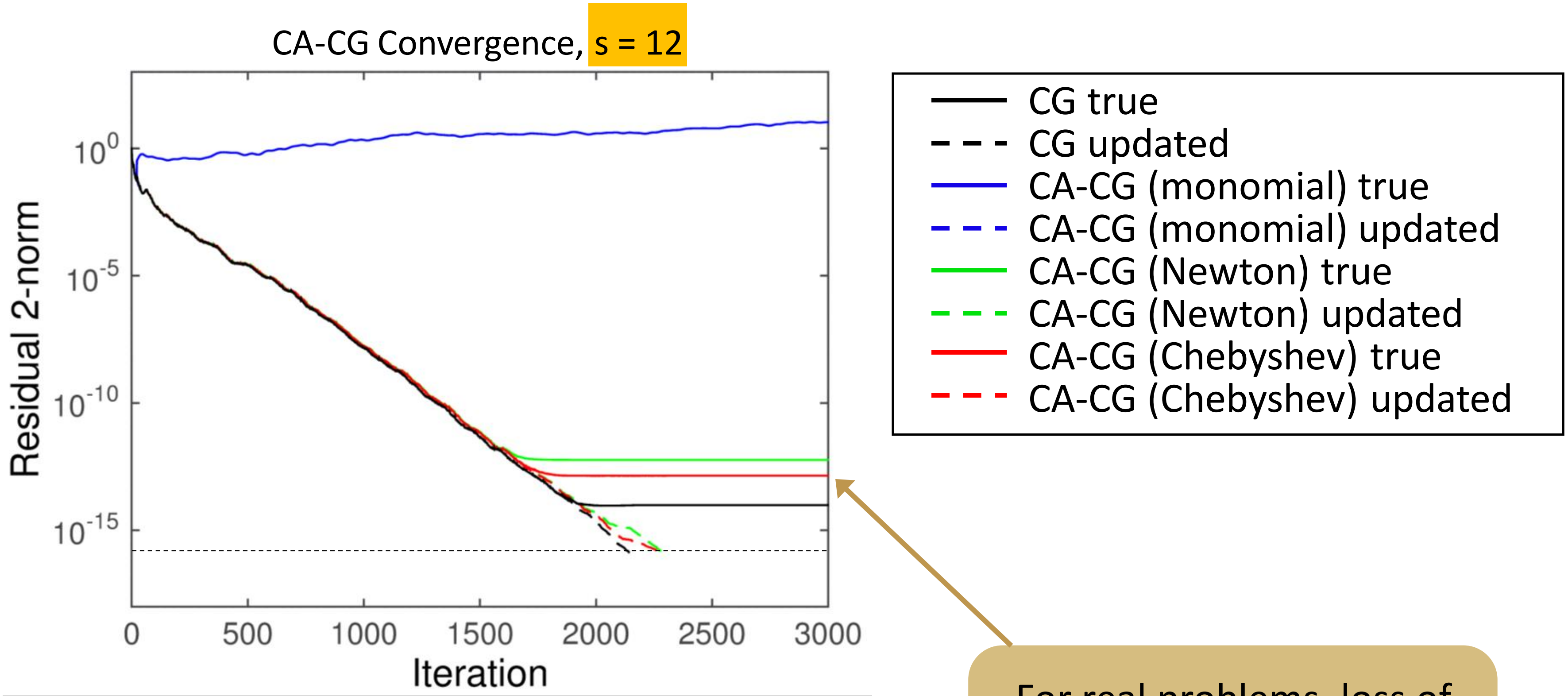
Better basis choice allows higher s values

CA-CG Convergence, s = 16

But can still see loss of accuracy

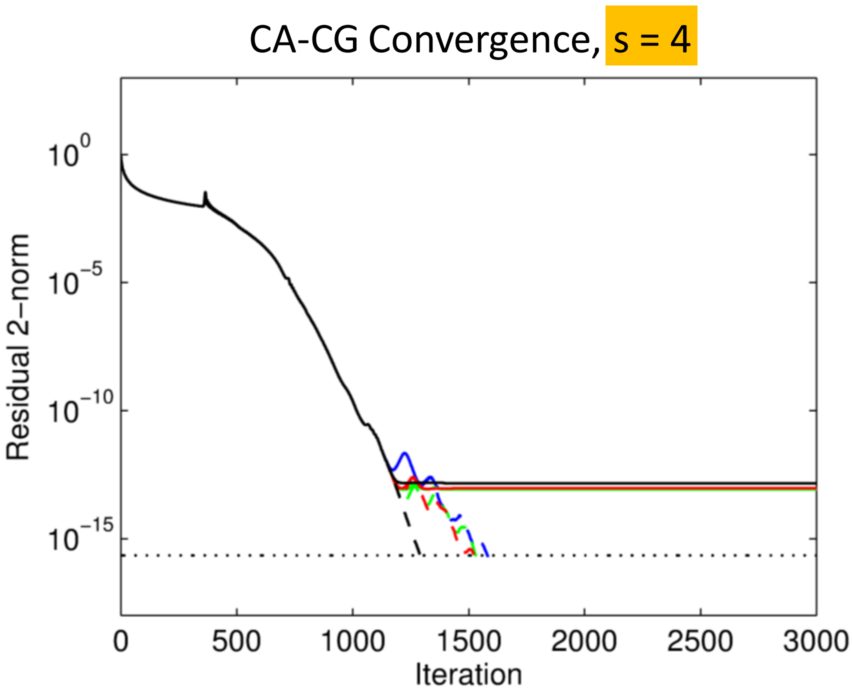
“consph” matrix (3D FEM), From UFL Sparse Matrix Collection

$$n = 8.3 \times 10^4,\ \mathrm{nnz} = 6.0 \times 10^6,\ \kappa(A) = 9.7 \times 10^3,\ \|A\|_2 = 9.7$$

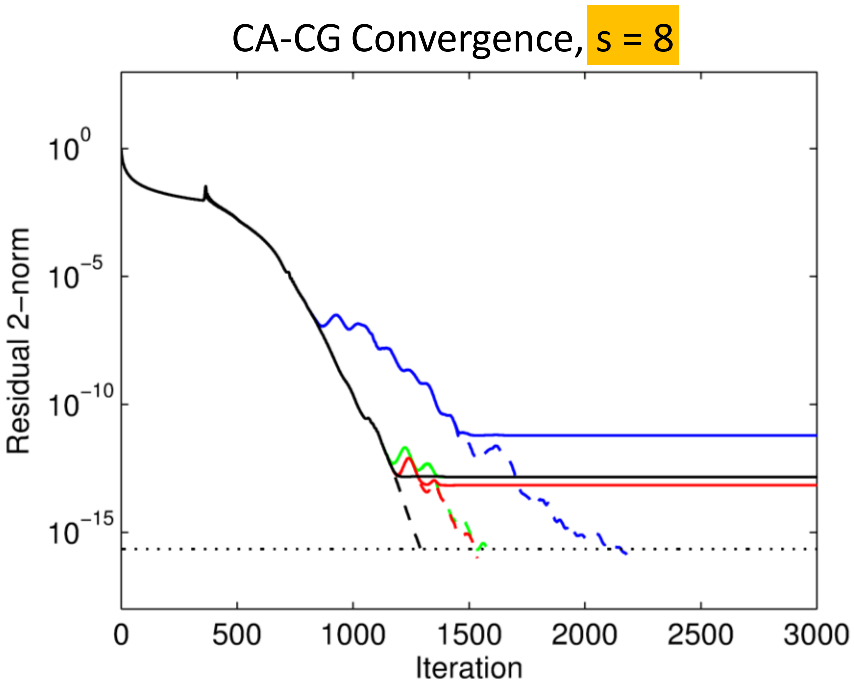


CA-CG Convergence, s = 12

For real problems, loss of accuracy becomes evident even with better bases



CA-CG Convergence, s = 4



CA-CG Convergence, s = 8

— CG true

- - - CG updated

— CA-CG (monomial) true

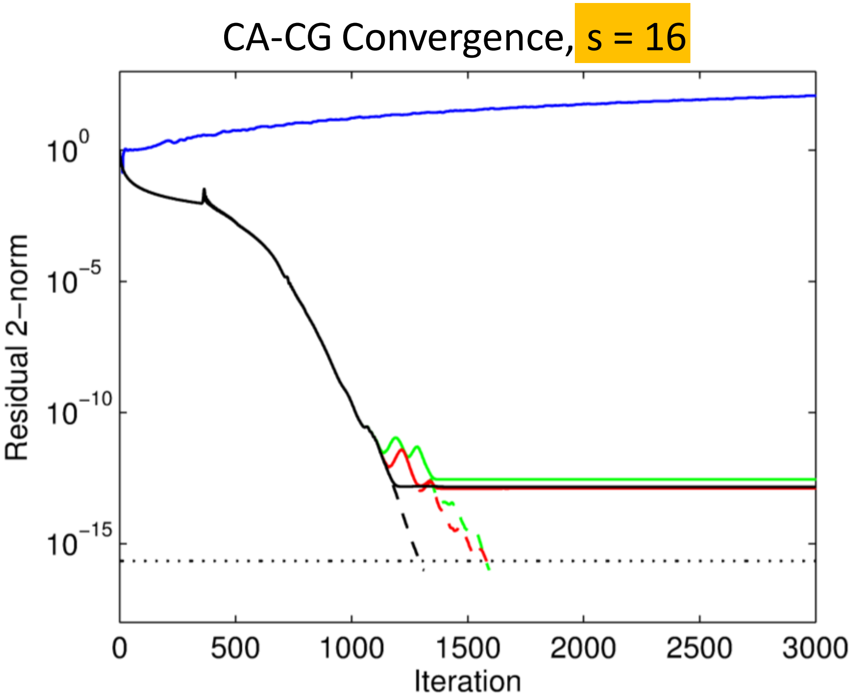
- - - CA-CG (monomial) updated

— CA-CG (Newton) true

- - - CA-CG (Newton) updated

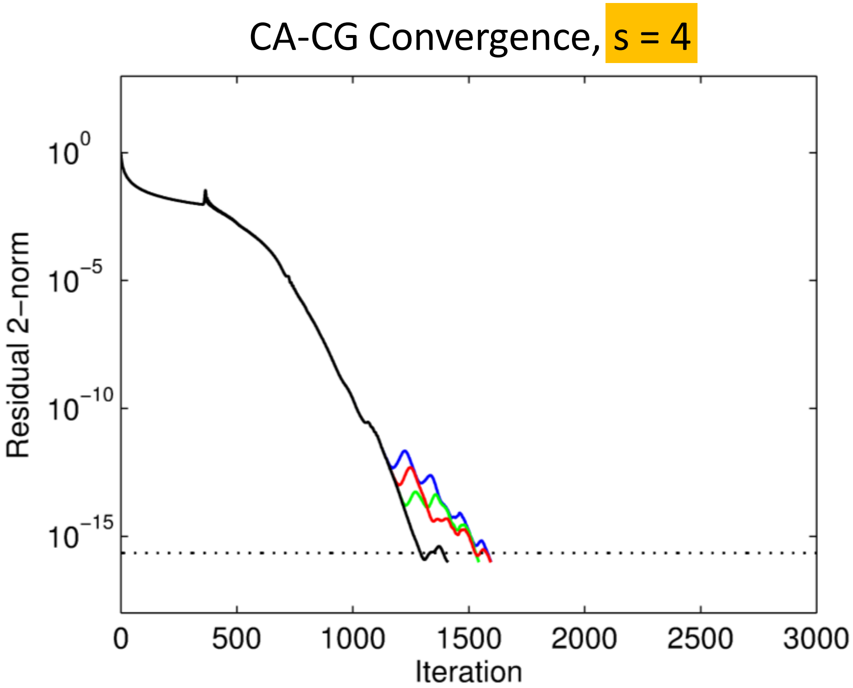
— CA-CG (Chebyshev) true

- - - CA-CG (Chebyshev) updated

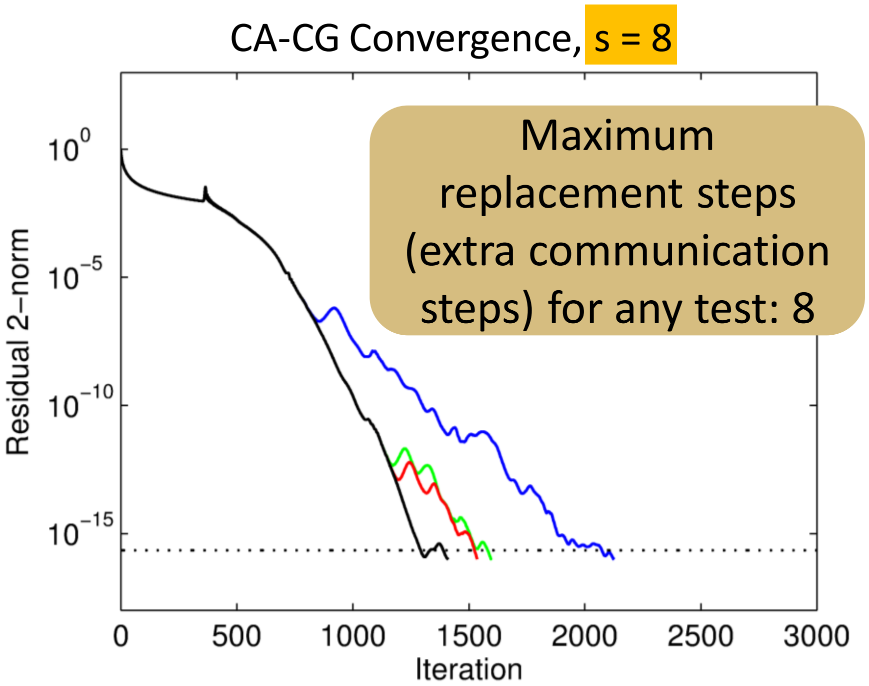


CA-CG Convergence, s = 16

Model Problem: 2D Poisson (5 pt stencil), n = 262K, nnz = 1.3M, cond(A) $\approx 10^4$



CA-CG Convergence, s = 4

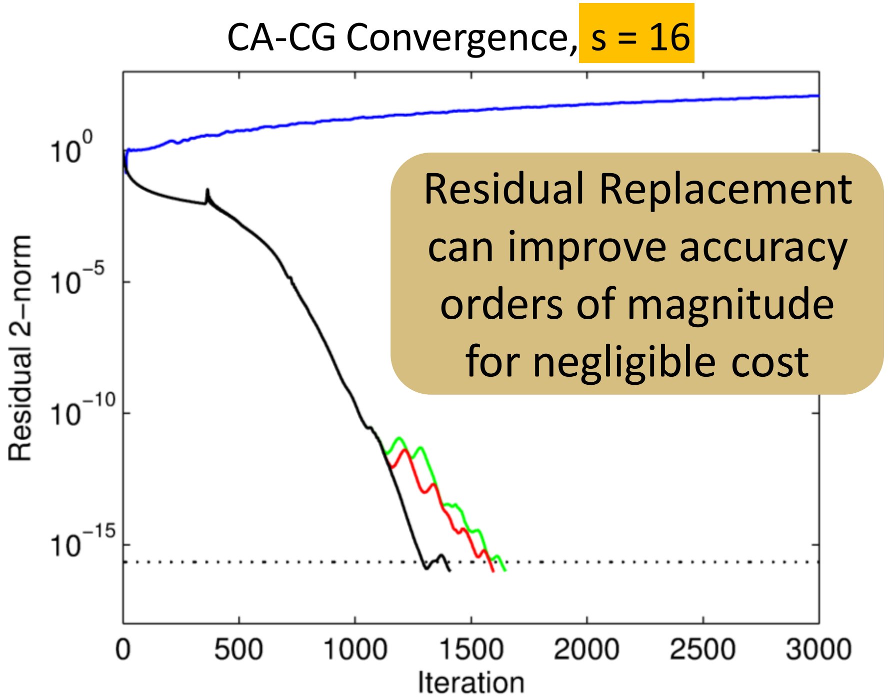


CA-CG Convergence, s = 8

Maximum replacement steps (extra communication steps) for any test: 8

———	CG-**RR** true
- - -	CG-**RR** updated
———	CA-CG-**RR** (monomial) true
- - -	CA-CG-**RR** (monomial) updated
———	CA-CG-**RR** (Newton) true
- - -	CA-CG-**RR** (Newton) updated
———	CA-CG-**RR** (Chebyshev) true
- - -	CA-CG-**RR** (Chebyshev) updated

Model Problem: 2D Poisson (5 pt stencil),
n = 262K, nnz = 1.3M, cond(A) ≈ 10^4



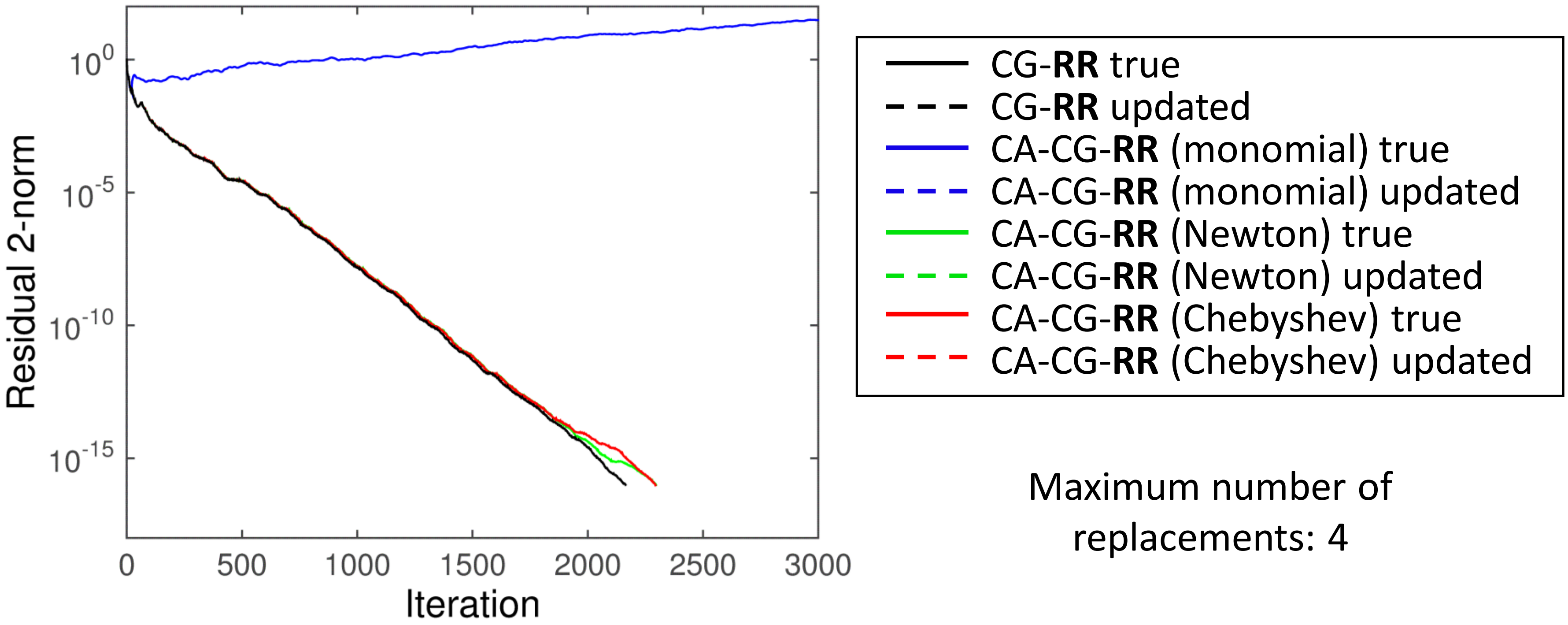
CA-CG Convergence, s = 16

Residual Replacement can improve accuracy orders of magnitude for negligible cost

“consph” matrix (3D FEM), From UFL Sparse Matrix Collection

$$n = 8.3 \times 10^4, \mathrm{nnz} = 6.0 \times 10^6, \kappa(A) = 9.7 \times 10^3, \|A\|_2 = 9.7$$

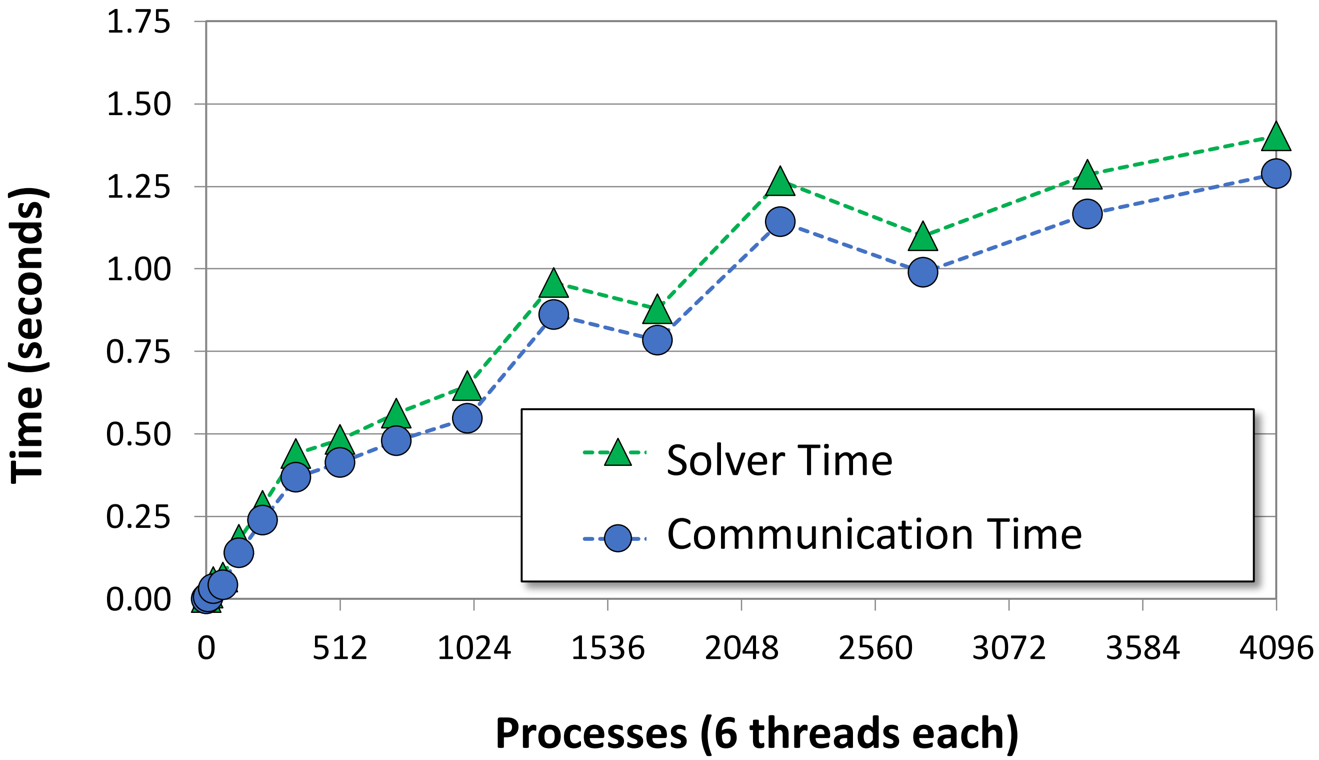
CA-CG Convergence, s = 12



Maximum number of replacements: 4

Coarse-grid Krylov Solver on NERSC's Hopper (Cray XE6)

Weak Scaling: 4^3 points per process (0 slope ideal)

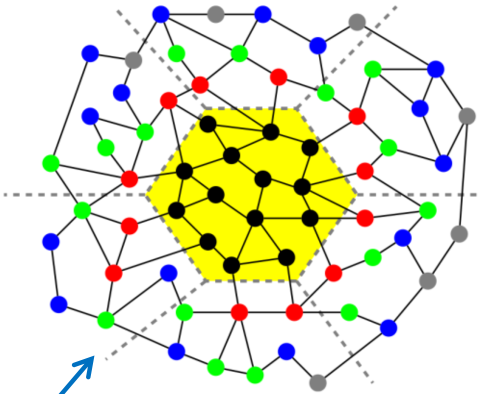


Solver performance and scalability limited by communication!

The Matrix Powers Kernel (Demmel et al., 2007)

Avoids communication:

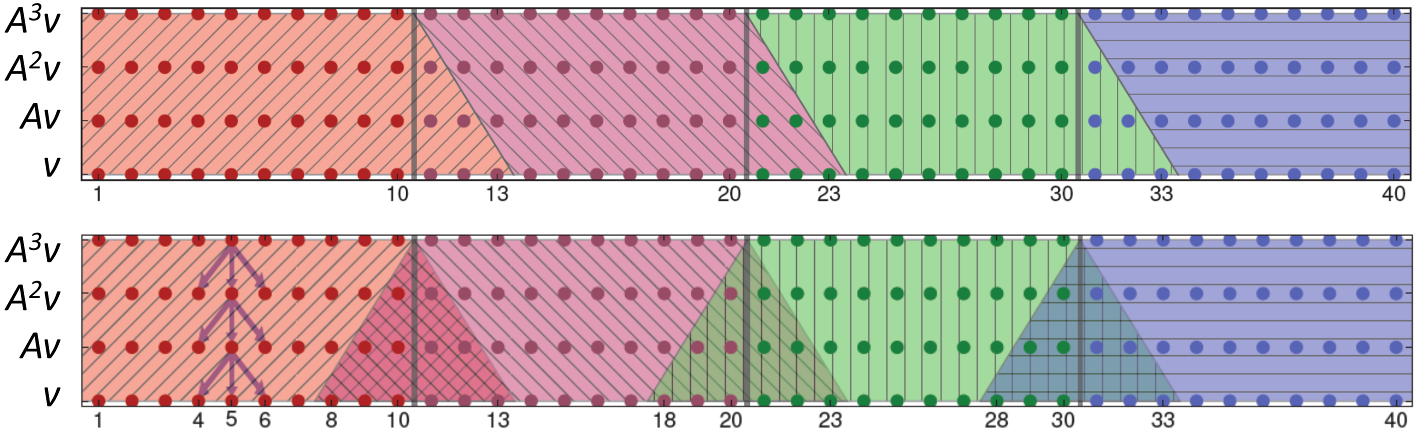
- In serial, by exploiting temporal locality:
  - Reading A, reading vectors
- In parallel, by doing only 1 ‘expand’ phase (instead of s).
- Requires sufficiently low ‘surface-to-volume’ ratio



Also works for general graphs!

black = local elements
red = 1-level dependencies
green = 2-level dependencies
blue = 3-level dependencies

Tridiagonal Example:



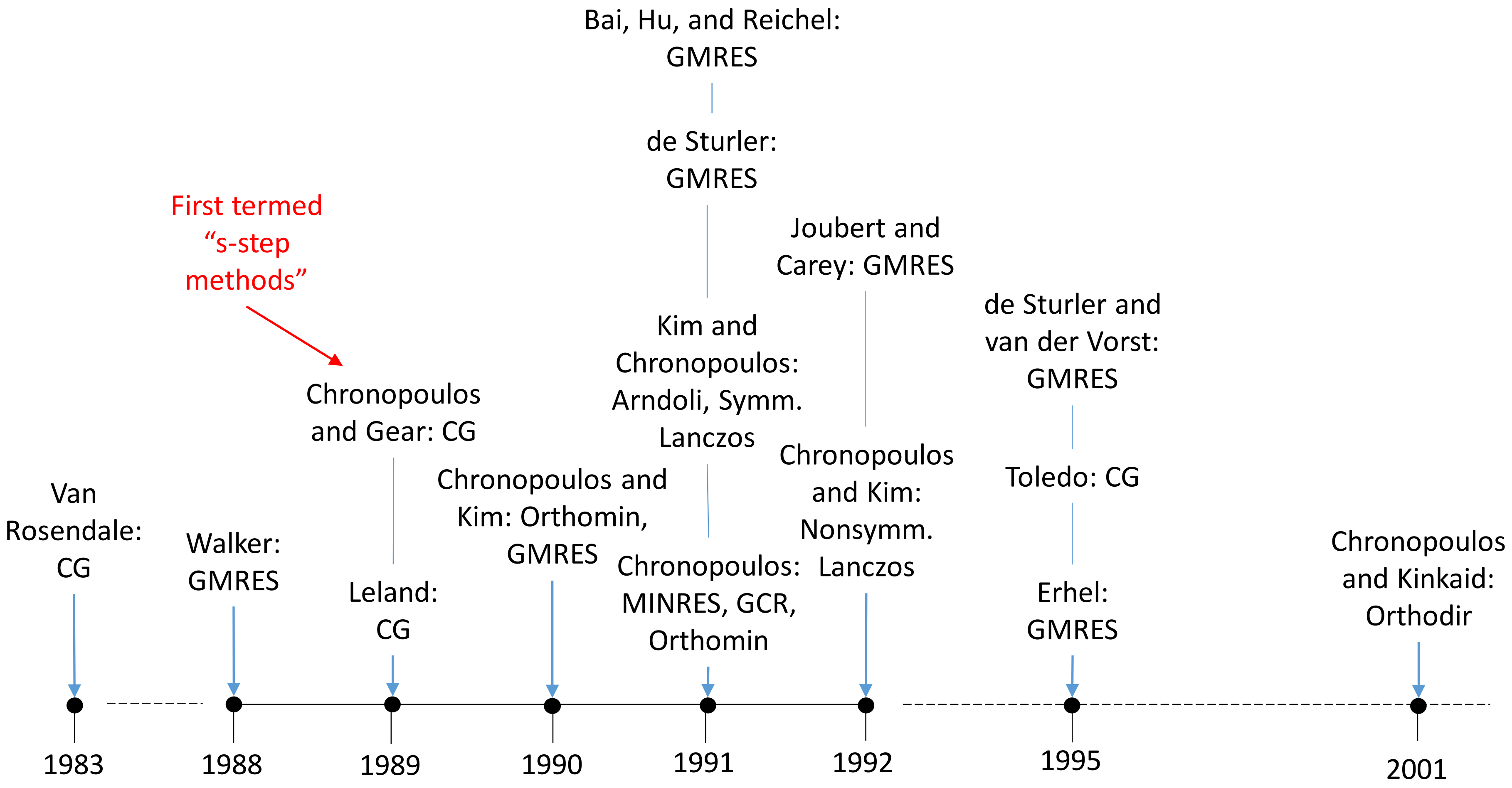
Sequential

Parallel

Choosing a Polynomial Basis

- Recall: in each outer loop of CA-CG, we compute bases for some Krylov subspaces, $\mathcal{K}_m(A, v) = \text{span}\{v, Av, \dots, A^{m-1}v\}$
- Simple loop unrolling gives monomial basis $Y = [p, Ap, A^2p, A^3p, \dots]$
  - Condition number can grow exponentially with s
    - Condition number = ratio of largest to smallest eigenvalues, $\lambda_{\max}/\lambda_{\min}$
  - Recognized early on that this negatively affects convergence (Leland, 1989)
- **Improve basis condition number to improve convergence**: Use different polynomials to compute a basis for the same subspace.
- Two choices based on spectral information that usually lead to well-conditioned bases:
  - **Newton polynomials**
  - **Chebyshev polynomials**

History of *s*-step Krylov Methods



1983
Van Rosendale: CG

1988
Walker: GMRES

1989
Chronopoulos and Gear: CG
First termed "s-step methods"
Leland: CG

1990
Chronopoulos and Kim: Orthomin, GMRES

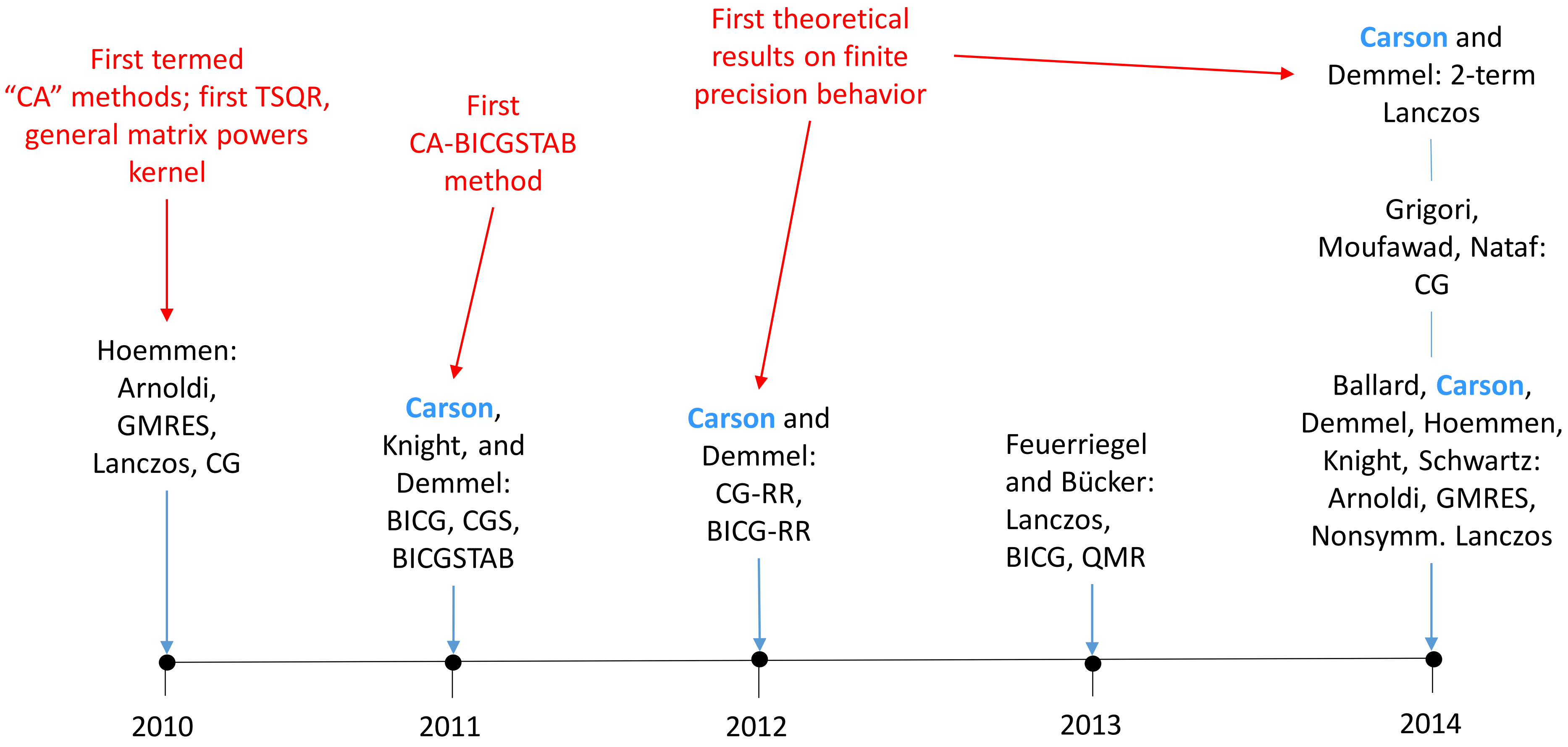
1991
Bai, Hu, and Reichel: GMRES
de Sturler: GMRES
Kim and Chronopoulos: Arndoli, Symm. Lanczos
Chronopoulos: MINRES, GCR, Orthomin

1992
Joubert and Carey: GMRES
Chronopoulos and Kim: Nonsymm. Lanczos

1995
de Sturler and van der Vorst: GMRES
Toledo: CG
Erhel: GMRES

2001
Chronopoulos and Kinkaid: Orthodir

Recent Years…



First termed "CA" methods; first TSQR, general matrix powers kernel

Hoemmen: Arnoldi, GMRES, Lanczos, CG

2010

First CA-BICGSTAB method

Carson, Knight, and Demmel: BICG, CGS, BICGSTAB

2011

First theoretical results on finite precision behavior

Carson and Demmel: CG-RR, BICG-RR

2012

Feuerriegel and Bücker: Lanczos, BICG, QMR

2013

Carson and Demmel: 2-term Lanczos

Grigori, Moufawad, Nataf: CG

Ballard, Carson, Demmel, Hoemmen, Knight, Schwartz: Arnoldi, GMRES, Nonsymm. Lanczos

2014

The Amplification Term Γ

- Roundoff errors in CA variant follow same pattern as classical variant, but amplified by factor of Γ or Γ^2
  - **Theoretically confirms observations** on importance of basis conditioning (dating back to late '80s)
- Need $\| |\mathcal{Y}||y'| \|_2 \leq \Gamma \|\mathcal{Y}y'\|_2$ to hold for the computed basis $\mathcal{Y}$ and coordinate vector y' in every bound.
- A loose bound for the amplification term:

$$\Gamma \leq \max_{\ell \leq k} \|\mathcal{Y}_\ell^+\|_2 \cdot \| |\mathcal{Y}_\ell| \|_2 \leq (2s+1) \cdot \max_{\ell \leq k} \kappa(\mathcal{Y}_\ell)$$

- What we really need: $\| |\mathcal{Y}||y'| \|_2 \leq \Gamma \|\mathcal{Y}y'\|_2$ to hold for the computed basis $\mathcal{Y}$ and coordinate vector y' in every bound.

- **Tighter bound on Γ possible**; requires some light bookkeeping
- Example:

$$\Gamma_{k,j} \equiv \max_{x \in \{\hat{w}'_{k,j}, \hat{u}'_{k,j}, \hat{v}'_{k,j}, \hat{v}'_{k,j-1}\}} \frac{\left\| |\hat{\mathcal{Y}}_k||x| \right\|_2}{\left\| \hat{\mathcal{Y}}_k x \right\|_2}$$

More Current Work

- 2.5D symmetric eigensolver (Solomonik et al.)
- Write-Avoiding algorithms (talk by Harsha Vardhan Simhadri in afternoon session)
- CA sparse RRLU (Grigori, Cayrols, Demmel)
- CA Parallel Sparse-Dense Matrix-Matrix Multiplication (Koanantakool et al.)
- Lower bounds for general programs that access arrays (talk by Nick Knight in afternoon session)
- CA Support Vector Machines (talk by Yang You)
- CA-RRQR (Demmel, Grigori, Gu, Xiang)
- CA-SBR (Ballard, Demmel, Knight)

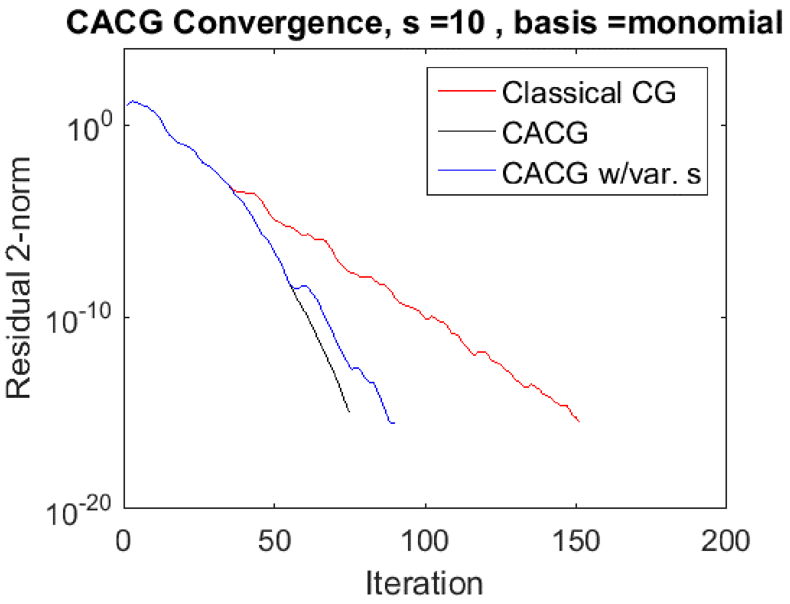
Dynamic basis size

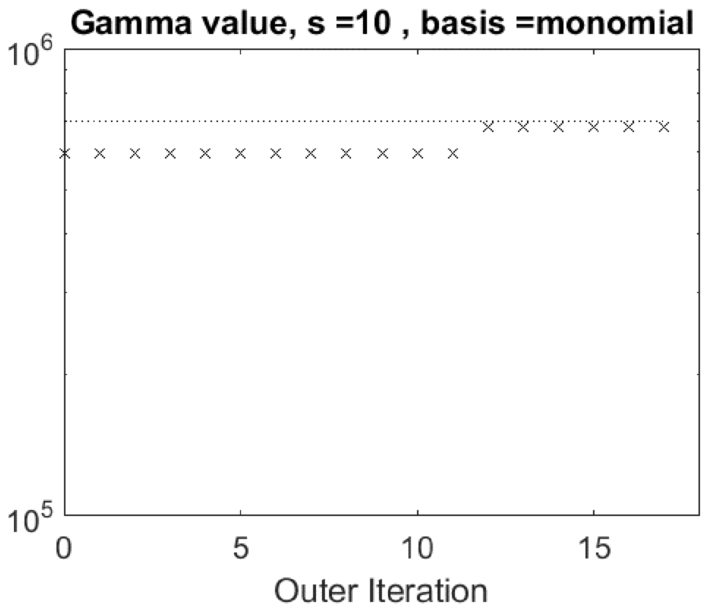
- Auto-tune to find best s based on machine, matrix sparsity structure; use this as $s_{\max}$
- In each outer iteration, select largest $s \leq s_{\max}$ such that

$$\kappa(\mathcal{Y}_k) \leq 1/\sqrt{\epsilon n}$$

- Benefit: Maintain acceptable convergence rate regardless of user's choice of s
- Cost: Incremental condition number estimation in each outer iteration; potentially wasted SpMVs in each outer iteration

s values used = (6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 4)





(CA-)Lanczos Convergence Analysis

Finite precision Lanczos process: (A is $n \times n$ with at most N nonzeros per row)

$$A\hat{V}_m = \hat{V}_m \hat{T}_m + \hat{\beta}_{m+1}\hat{v}_{m+1}e_m^T + \delta\hat{V}_m$$

$$\hat{V}_m = [\hat{v}_1, \ldots, \hat{v}_m], \qquad \delta\hat{V}_m = [\delta\hat{v}_1, \ldots, \delta\hat{v}_m], \qquad \hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & & \\ \hat{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \hat{\beta}_m \\ & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

for $i \in \{1, \ldots, m\}$,

$$\|\delta\hat{v}_i\|_2 \leq \varepsilon_1\sigma$$
$$\hat{\beta}_{i+1}\left|\hat{v}_i^T\hat{v}_{i+1}\right| \leq 2\varepsilon_0\sigma$$
$$\left|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1\right| \leq \varepsilon_0/2$$
$$\left|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2\right| \leq 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2$$

where $\sigma \equiv \|A\|_2$, and
$\theta\sigma \equiv \||A|\|_2$

Classical Lanczos (Paige, 1976):

$$\varepsilon_0 = O(\varepsilon n)$$
$$\varepsilon_1 = O(\varepsilon N\theta)$$

CA-Lanczos (C., 2015):

$$\varepsilon_0 = O\left(\varepsilon n \boldsymbol{\Gamma}^{\mathbf{2}}\right)$$
$$\varepsilon_1 = O(\varepsilon N\theta\boldsymbol{\Gamma})$$

$$\Gamma = \max_{\ell \leq k} \|\mathcal{Y}_\ell^+\|_2 \cdot \||\mathcal{Y}_\ell|\|_2$$

Paige’s Results for Classical Lanczos (1980)

- Using bounds on local rounding errors in Lanczos, showed that
  1. The computed eigenvalues always lie between the extreme eigenvalues of A to within a small multiple of machine precision.
  2. At least one small interval containing an eigenvalue of A is found by the nth iteration.
  3. The algorithm behaves numerically like Lanczos with full reorthogonalization until a very close eigenvalue approximation is found.
  4. The loss of orthogonality among basis vectors follows a rigorous pattern and implies that some computed eigenvalues have converged.

Do the same statements hold for CA-Lanczos?

Paige’s Lanczos Convergence Analysis

$$A\hat{V}_m = \hat{V}_m\hat{T}_m + \hat{\beta}_{m+1}\hat{v}_{m+1}e_m^T + \delta\hat{V}_m$$

$$\hat{V}_m = [\hat{v}_1, \dots, \hat{v}_m], \qquad \delta\hat{V}_m = [\delta\hat{v}_1, \dots, \delta\hat{v}_m], \qquad \hat{T}_m = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_2 & & \\ \hat{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \hat{\beta}_m \\ & & \hat{\beta}_m & \hat{\alpha}_m \end{bmatrix}$$

Classic Lanczos rounding error result of Paige (1976):

for $i \in \{1, \dots, m\}$,

$$\|\delta\hat{v}_i\|_2 \leq \varepsilon_1\sigma$$

$$\hat{\beta}_{i+1}\left|\hat{v}_i^T\hat{v}_{i+1}\right| \leq 2\varepsilon_0\sigma$$

$$\left|\hat{v}_{i+1}^T\hat{v}_{i+1} - 1\right| \leq \varepsilon_0/2$$

$$\left|\hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2\right| \leq 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2$$

where $\sigma \equiv \|A\|_2$, $\theta\sigma \equiv \||A|\|_2$, $\varepsilon_0 \equiv 2\varepsilon(n+4)$, and $\varepsilon_1 \equiv 2\varepsilon(N\theta + 7)$

$\uparrow$ $\varepsilon_0 = O(\varepsilon n)$ $\qquad$ $\uparrow$ $\varepsilon_1 = O(\varepsilon N\theta)$

→ These results form the basis for Paige’s influential results in (Paige, 1980).

CA-Lanczos Convergence Analysis

Let $\Gamma \equiv \max_{\ell \le k} \|Y_\ell^+\|_2 \cdot \||Y_\ell|\|_2 \le (2s+1) \cdot \max_{\ell \le k} \kappa(Y_\ell)$.

For CA-Lanczos, we have:

for $i \in \{1, \ldots, m{=}sk{+}j\}$,

$$\begin{aligned} \|\delta \hat{v}_i\|_2 &\le \varepsilon_1 \sigma \\ \hat{\beta}_{i+1} \left| \hat{v}_i^T \hat{v}_{i+1} \right| &\le 2\varepsilon_0 \sigma \\ \left| \hat{v}_{i+1}^T \hat{v}_{i+1} - 1 \right| &\le \varepsilon_0 / 2 \\ \left| \hat{\beta}_{i+1}^2 + \hat{\alpha}_i^2 + \hat{\beta}_i^2 - \|A\hat{v}_i\|_2^2 \right| &\le 4i(3\varepsilon_0 + \varepsilon_1)\sigma^2 \end{aligned}$$

$\varepsilon_0 \equiv 2\varepsilon(n{+}11s{+}15)\, \Gamma^2 = O(\varepsilon n \Gamma^2)$, ← (vs. $O(\varepsilon n)$ for Lanczos)

$\varepsilon_1 \equiv 2\varepsilon\big((\mathrm{N}{+}2s{+}5)\theta + (4s{+}9)\tau + 10s{+}16\big)\Gamma = O(\varepsilon N \theta \Gamma)$, ← (vs. $O(\varepsilon N \theta)$ for Lanczos)

where $\sigma \equiv \|A\|_2$, $\theta\sigma \equiv \||A|\|_2$, $\tau\sigma \equiv \max_{\ell \le k} \||B_\ell|\|_2$

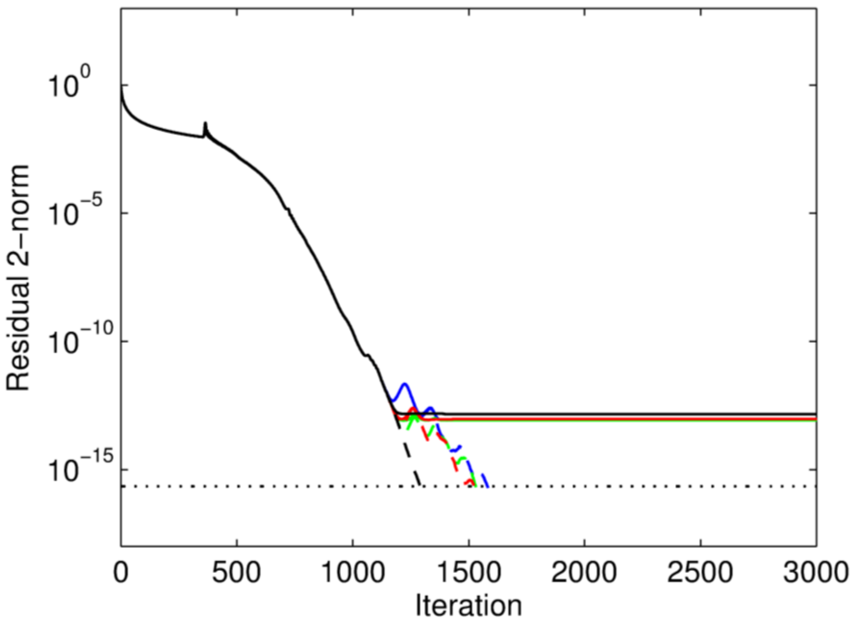
Residual Replacement Strategy

- van der Vorst and Ye (1999): improve accuracy by **replacing updated residual r_m by the true residual $b - Ax_m$** in certain iterations

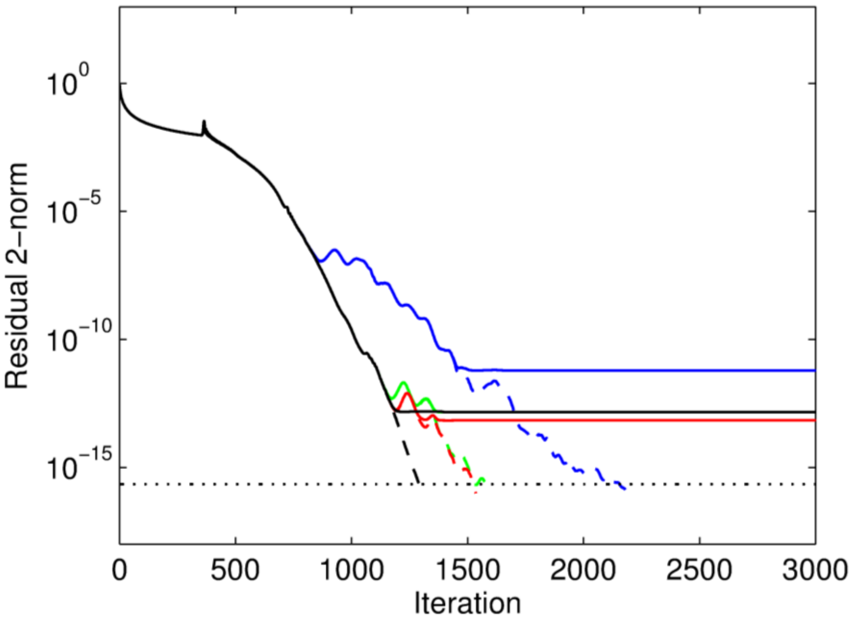
- Choose when to replace r_m with $b - Ax_m$ to meet two constraints:
  1. $||b - Ax_m - r_m||$ is small
  2. Convergence rate is maintained (avoid large perturbations to finite precision CG recurrence)

- Requires monitoring estimate of deviation of residuals

- We can use the same strategy for CA-CG
- Implementation has **negligible cost** → residual replacement strategy can allow **both speed and accuracy!**

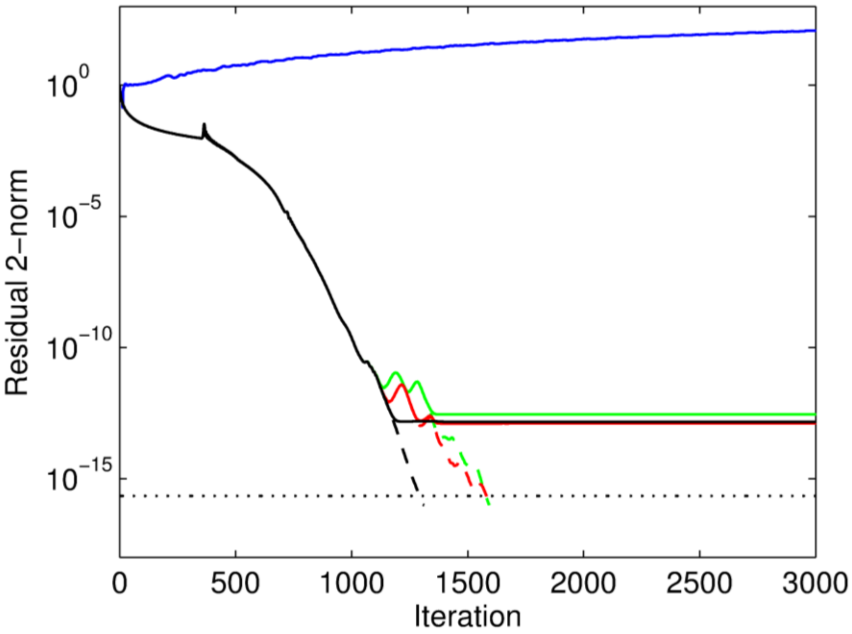


CA-CG Convergence, s = 4



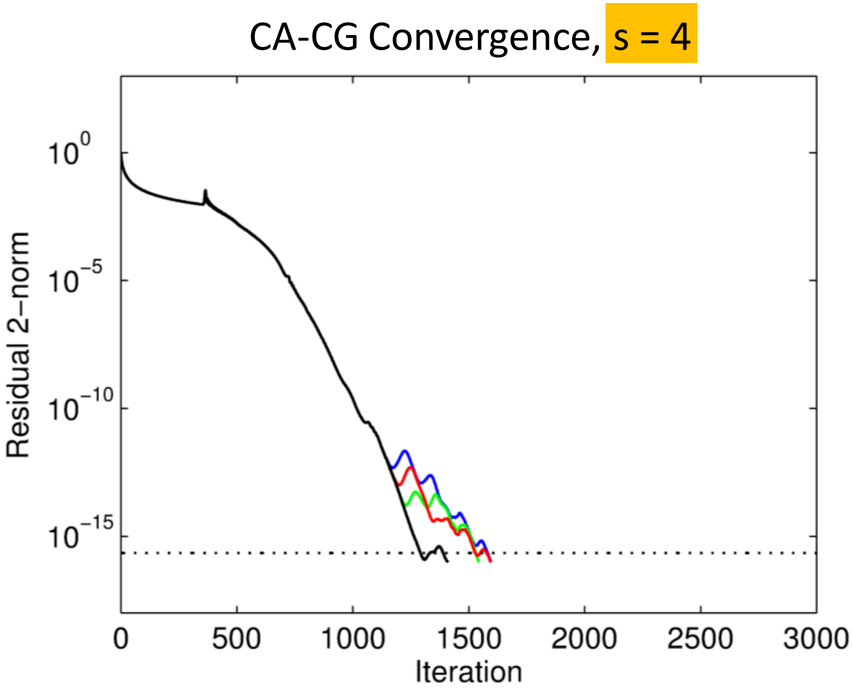
CA-CG Convergence, s = 8

—— CG true
- - - CG updated
—— CA-CG (monomial) true
- - - CA-CG (monomial) updated
—— CA-CG (Newton) true
- - - CA-CG (Newton) updated
—— CA-CG (Chebyshev) true
- - - CA-CG (Chebyshev) updated

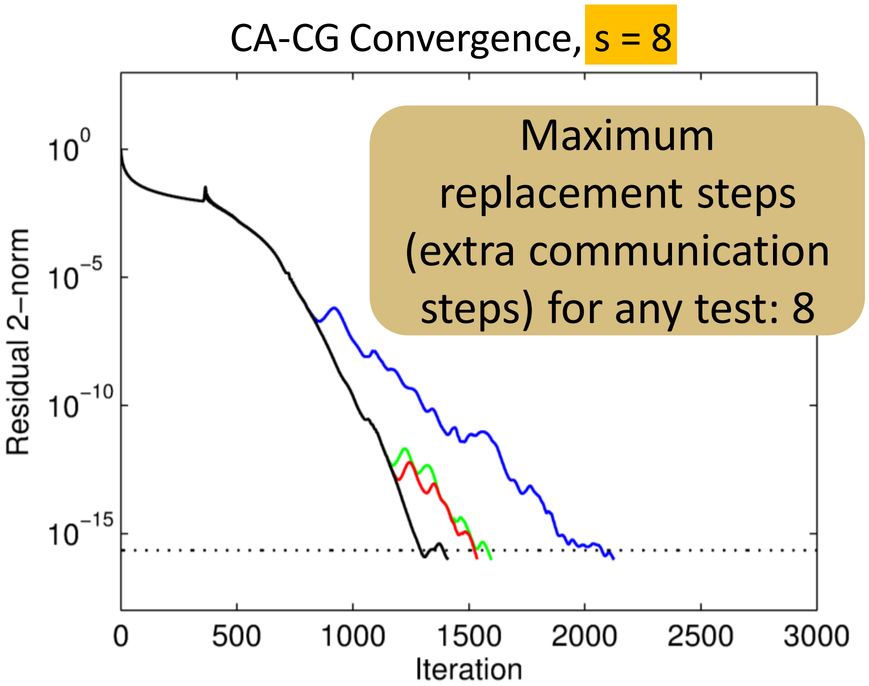


CA-CG Convergence, s = 16

Model Problem: 2D Poisson (5 pt stencil),
n = 262K, nnz = 1.3M, cond(A) ≈ 10^4



CA-CG Convergence, s = 4



CA-CG Convergence, s = 8

Maximum replacement steps (extra communication steps) for any test: 8

——— CG-**RR** true

- - - CG-**RR** updated

——— CA-CG-**RR** (monomial) true

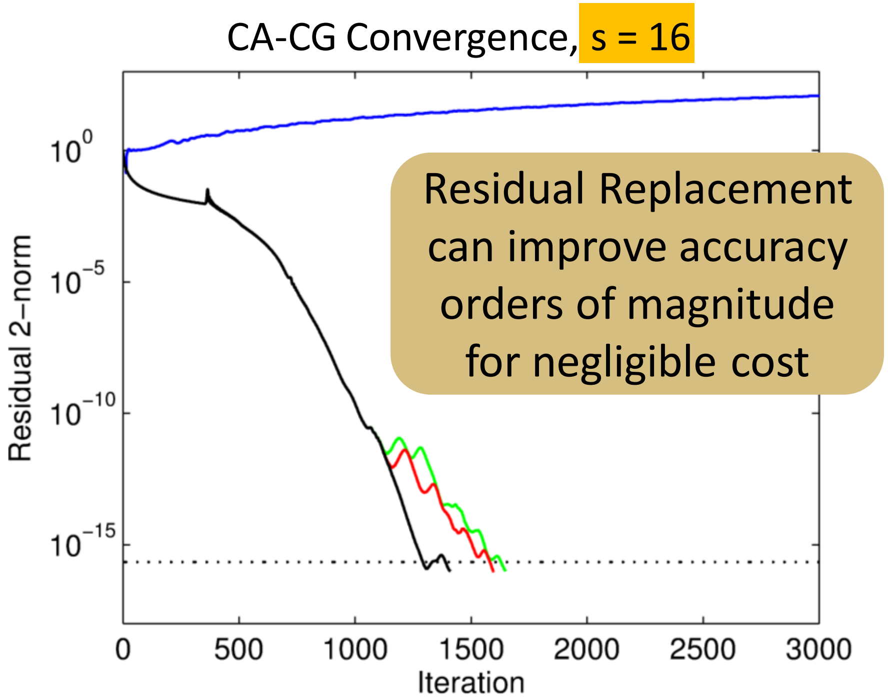
- - - CA-CG-**RR** (monomial) updated

——— CA-CG-**RR** (Newton) true

- - - CA-CG-**RR** (Newton) updated

——— CA-CG-**RR** (Chebyshev) true

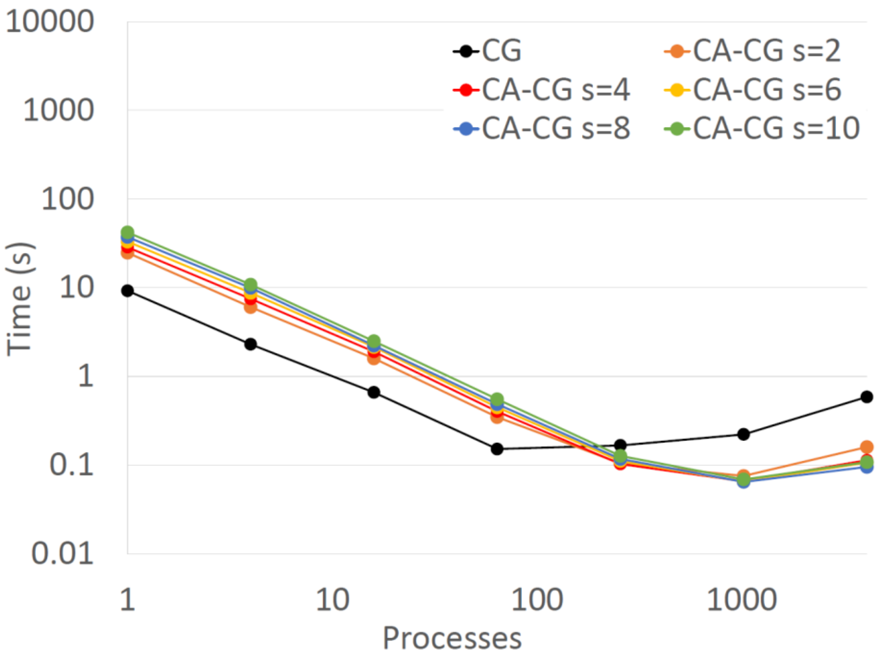
- - - CA-CG-**RR** (Chebyshev) updated

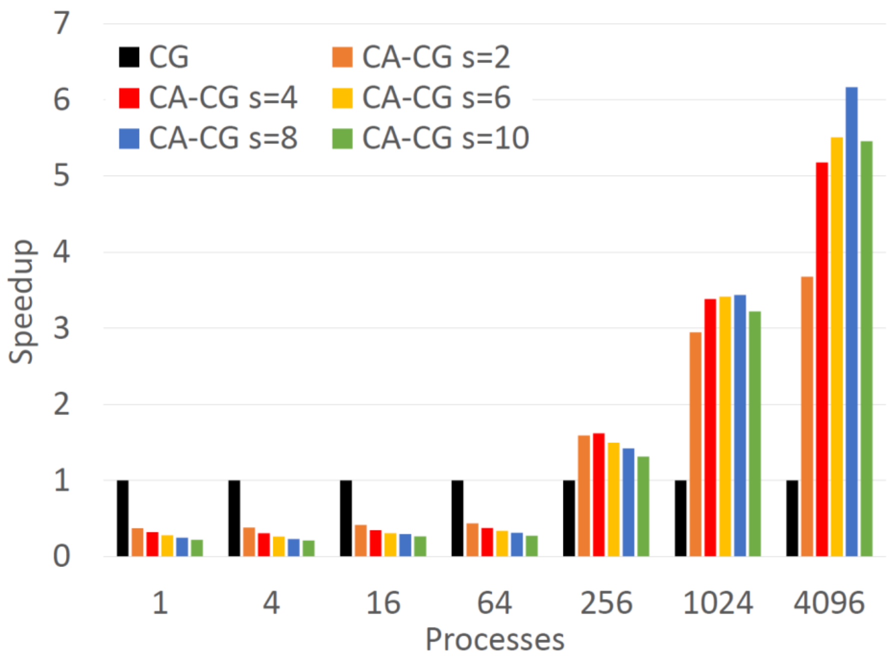


CA-CG Convergence, s = 16

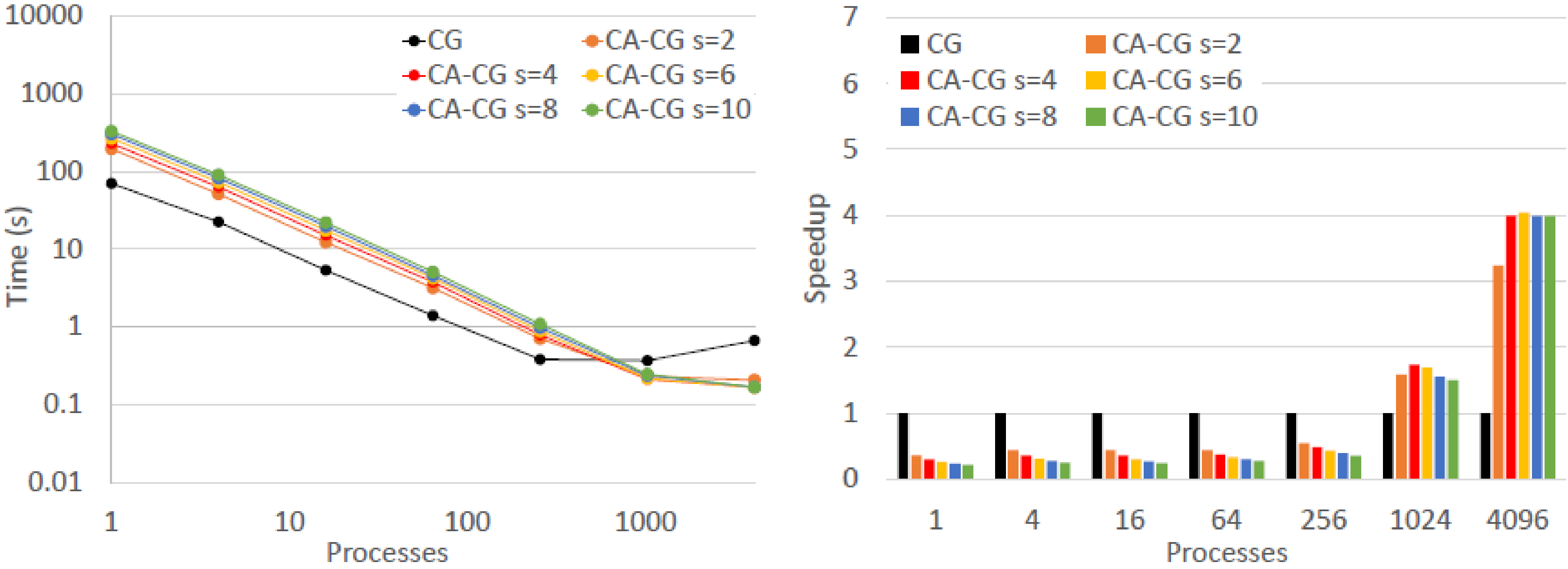
Residual Replacement can improve accuracy orders of magnitude for negligible cost

Model Problem: 2D Poisson (5 pt stencil),
n = 262K, nnz = 1.3M, cond(A) ≈ 10^4

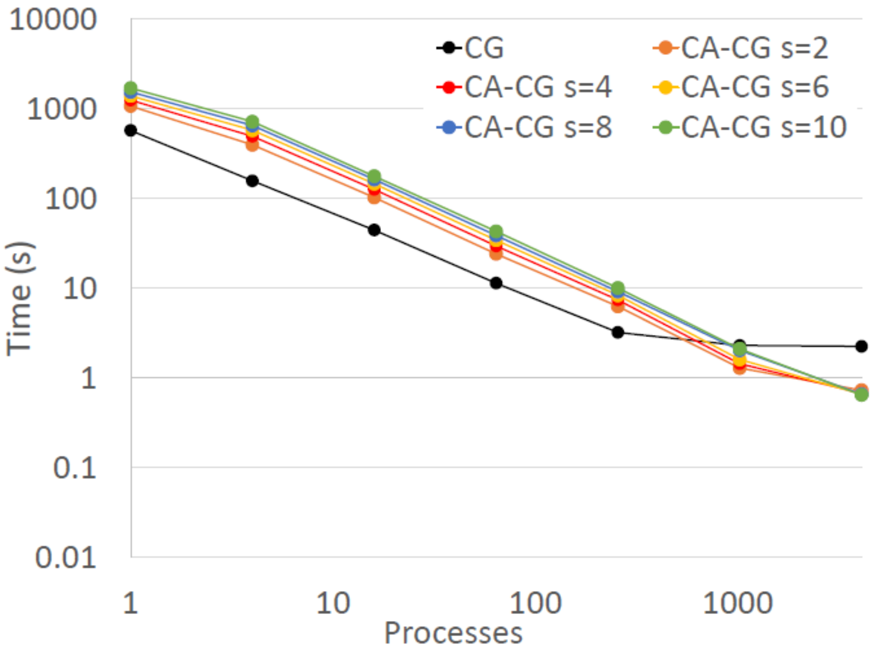


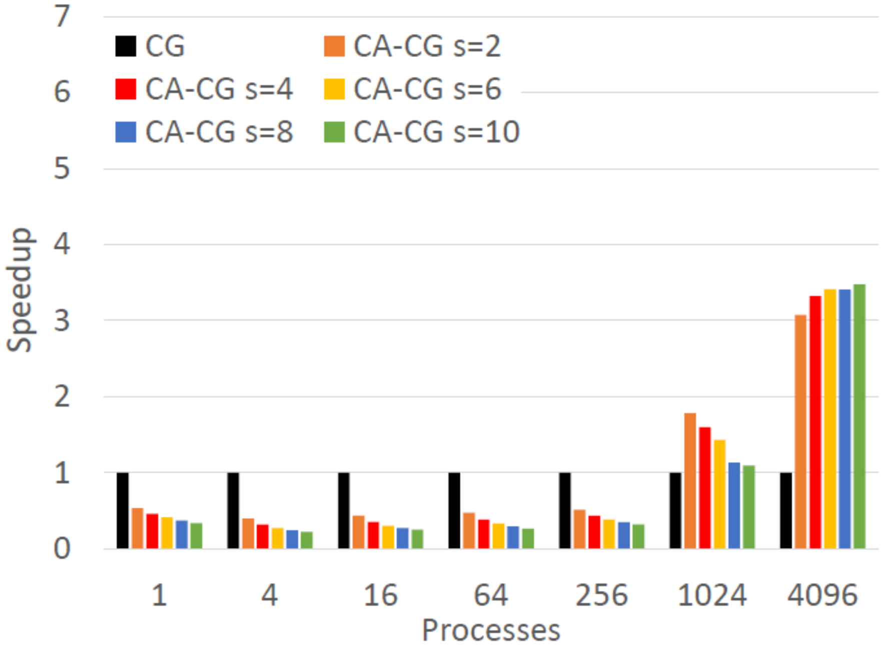


Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 512^2 grid

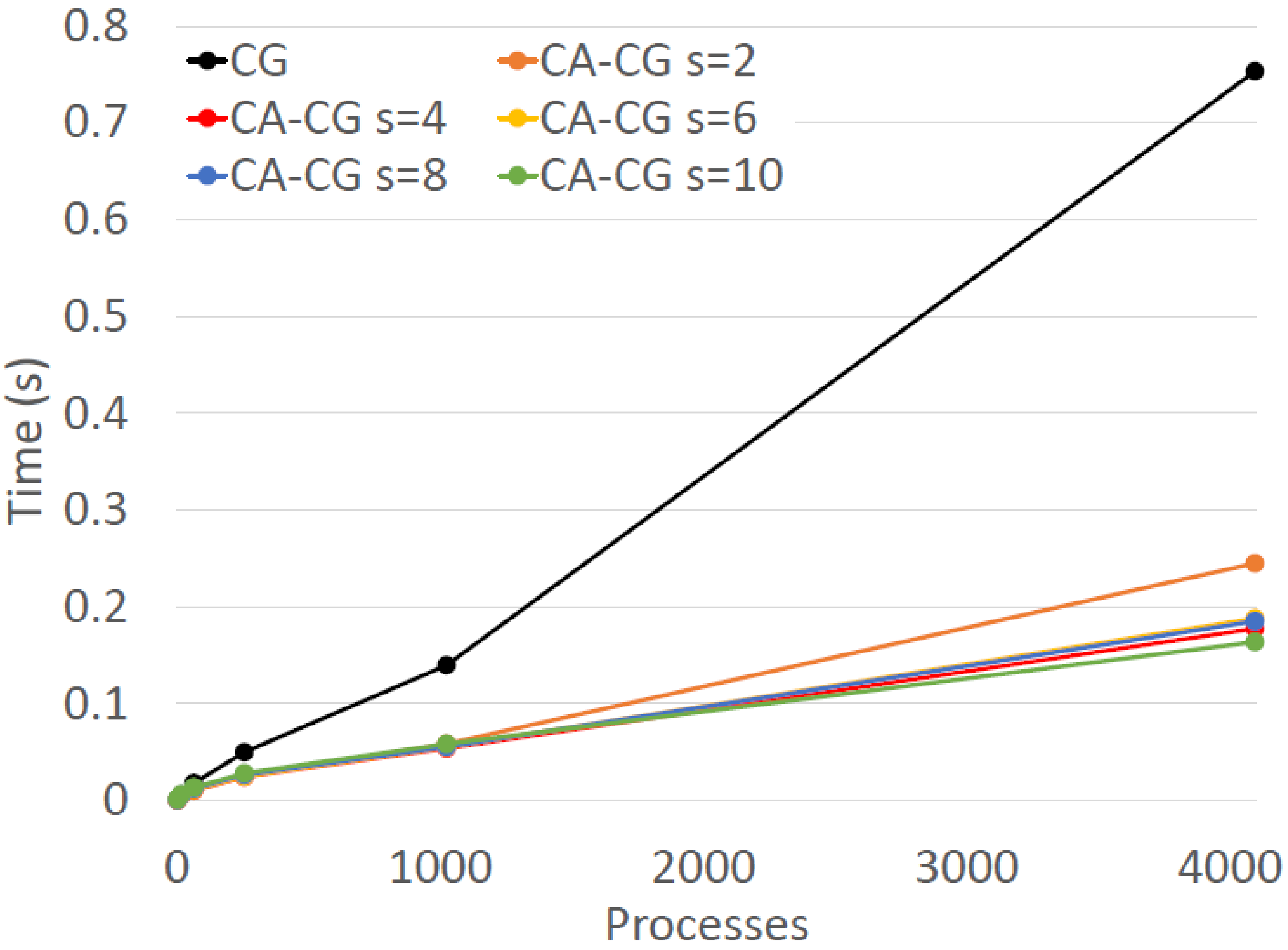


Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 1024^2 grid

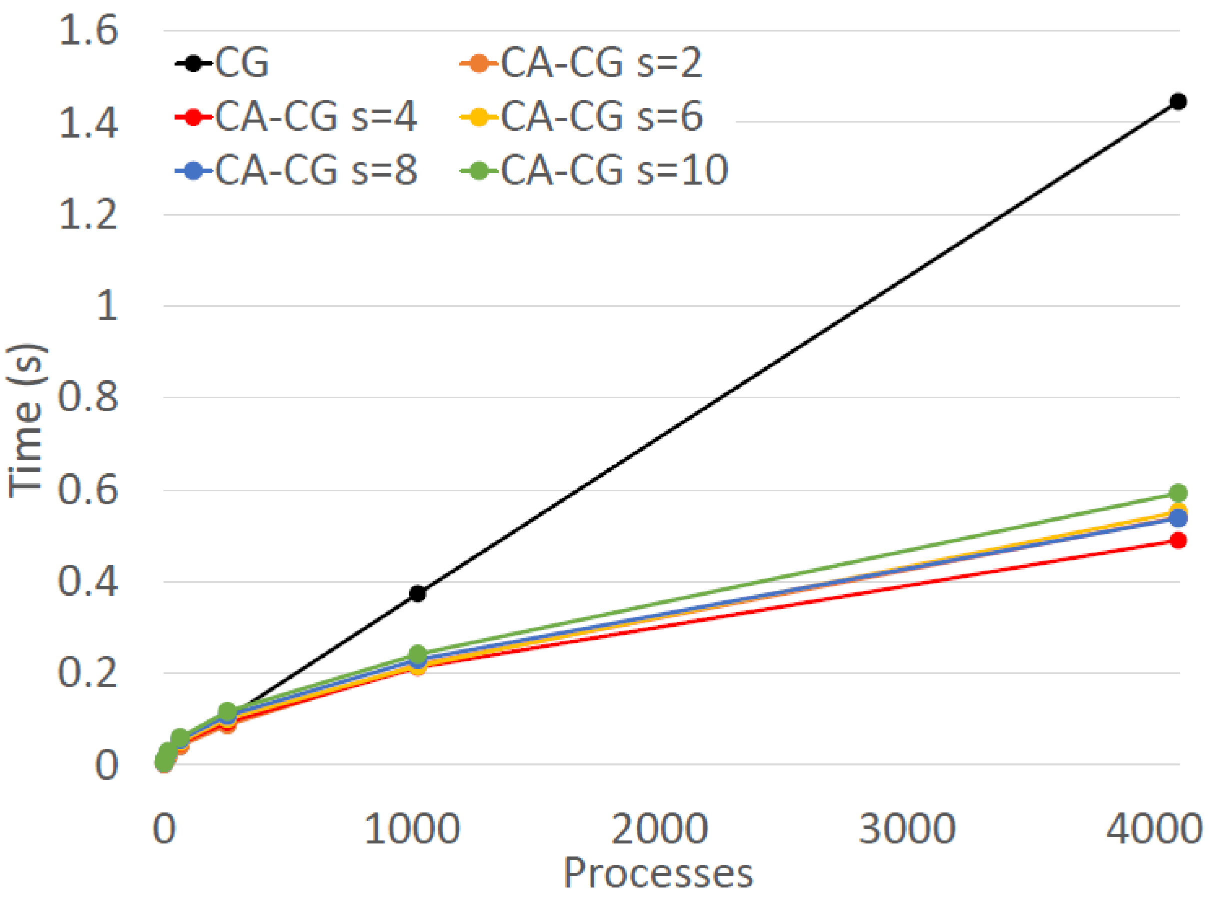




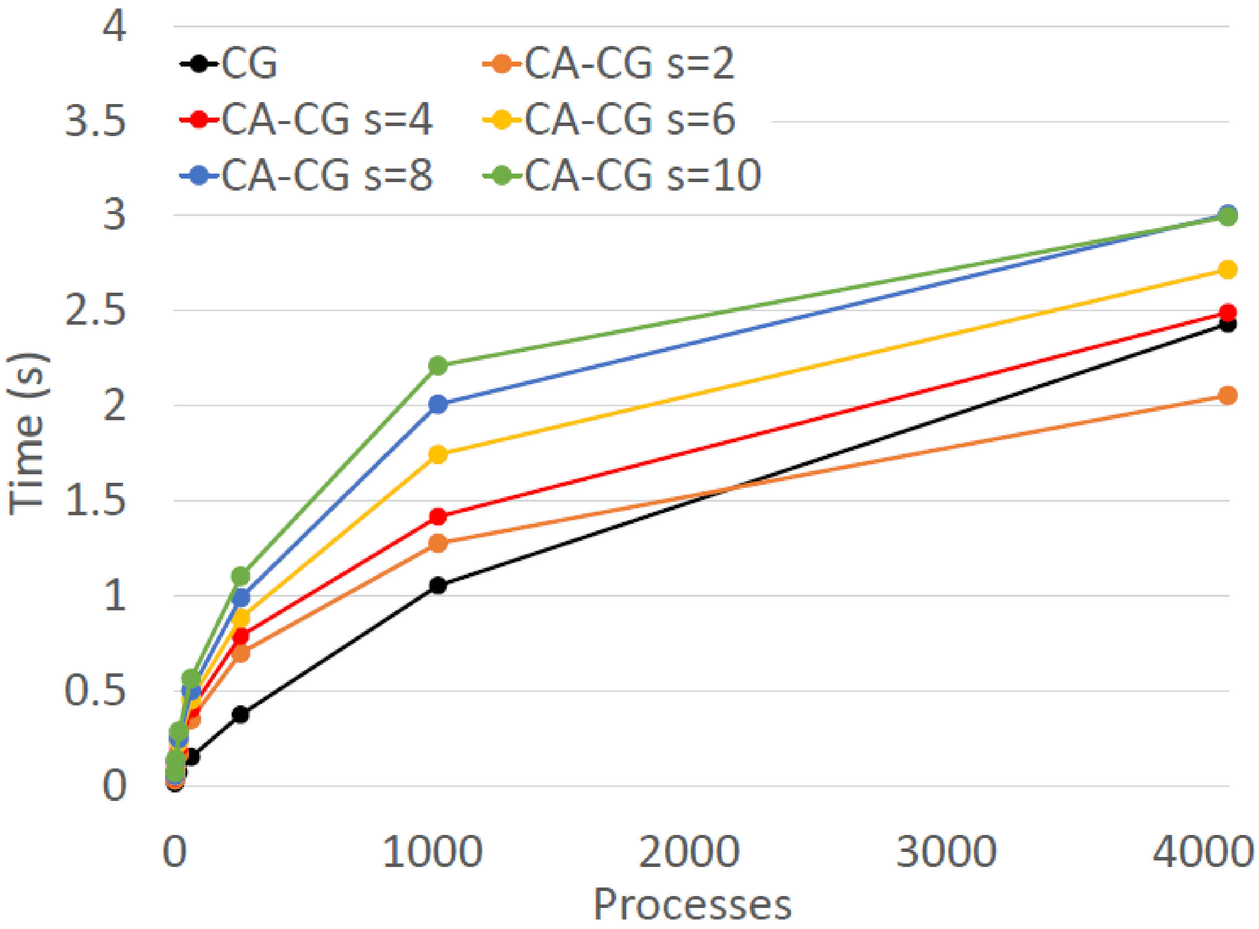
Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 2048^2 grid



Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 16^2 grid per process



Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 32^2 grid per process



Hopper, 4 MPI Processes per node
CG is PETSc solver
2D Poisson on 64^2 grid per process

Communication-Avoiding Krylov Method Speedups

- Recent results: CA-BICGSTAB used as geometric multigrid (GMG) bottom-solve (Williams, Carson, et al., IPDPS '14)
- Plot: Net time spent on different operations over one GMG bottom solve using 24,576 cores, 64^3 points/core on fine grid, 4^3 points/core on coarse grid
- Hopper at NERSC (Cray XE6), 4 6-core Opteron chips per node, Gemini network, 3D torus
- 3D Helmholtz equation

$$a\alpha u - b\nabla \cdot \beta \nabla u = f$$

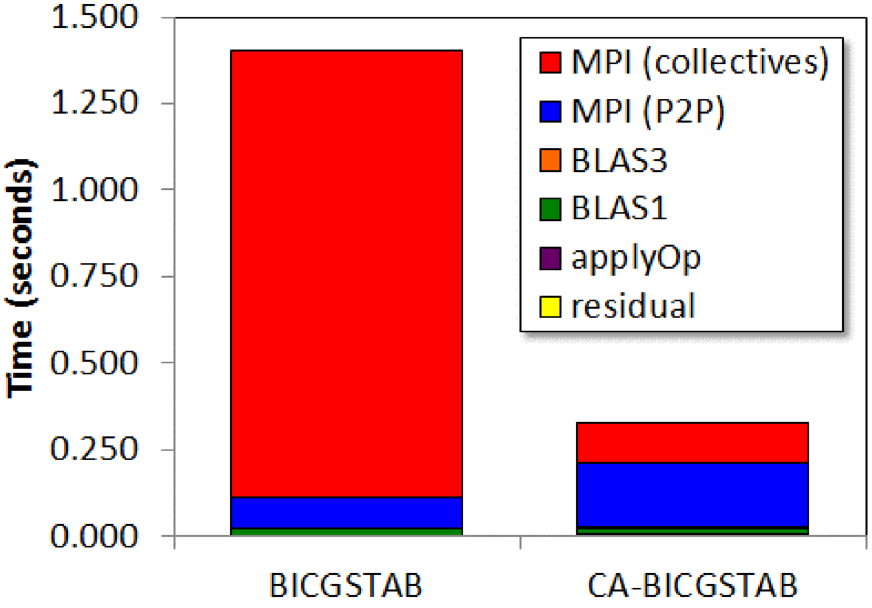
$$\alpha = \beta = 1.0,\ a = b = 0.9$$

- **CA-BICGSTAB with $s = 4$**

  4.2x speedup in Krylov solve;

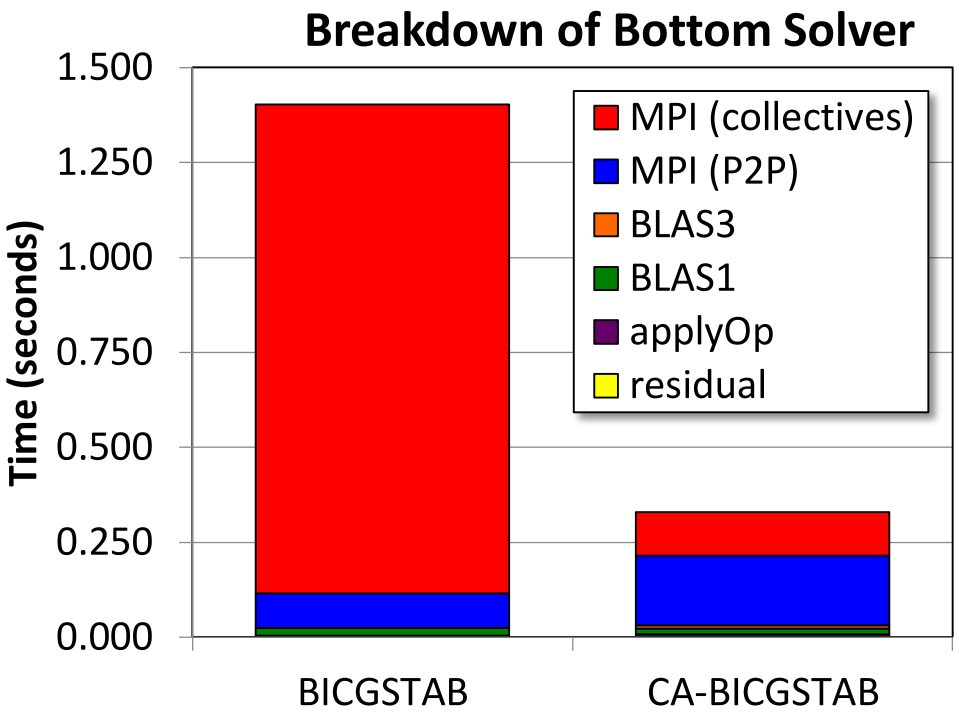
  2.5x in overall GMG solve

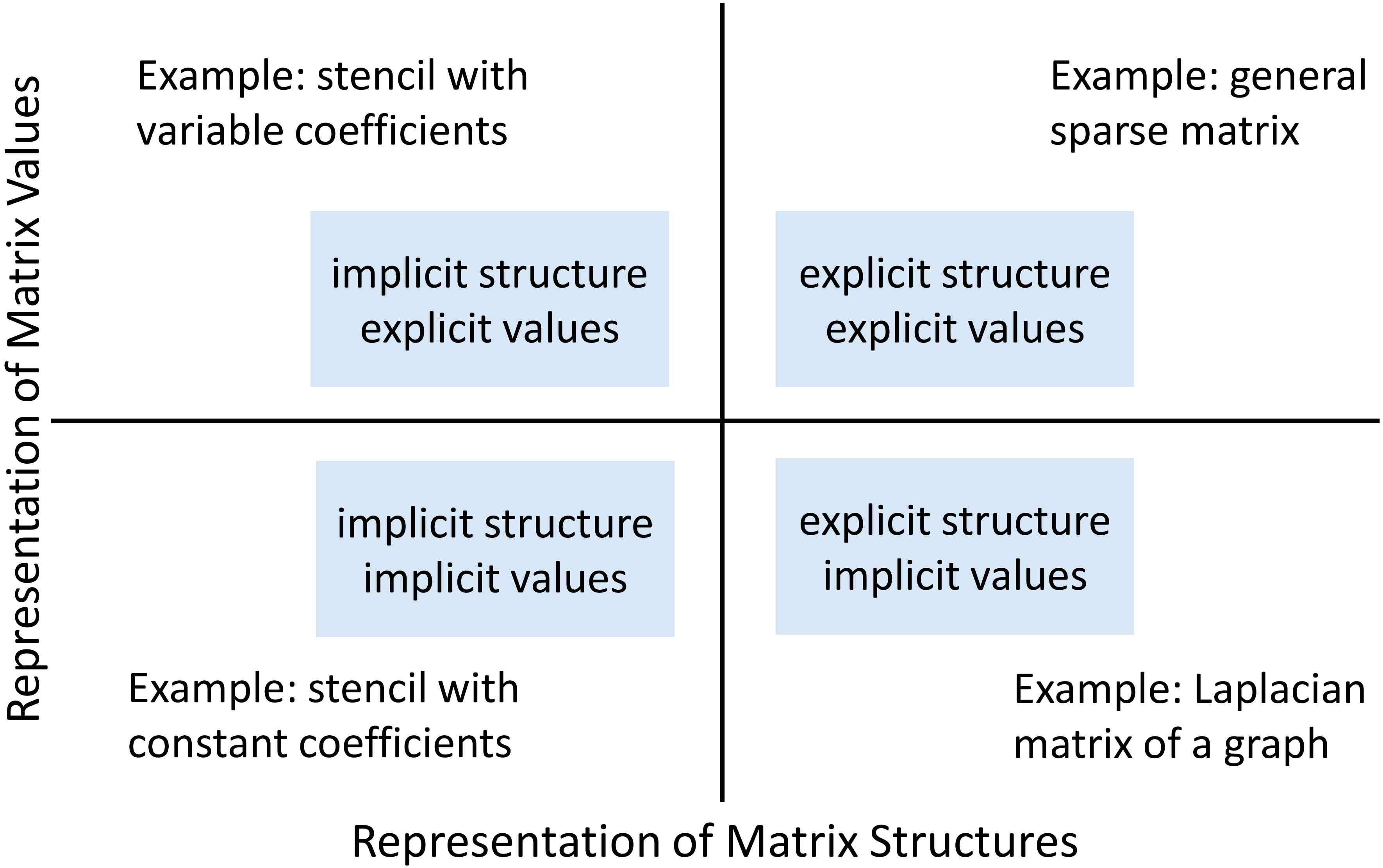
- Implemented in BoxLib: applied to low-Mach number combustion and 3D N-body dark matter simulation apps



Benchmark timing breakdown

- Plot: Net time spent across all bottom solves at $24{,}576$ cores, for BICGSTAB and CA-BICGSTAB with $s = 4$
- **11.2x reduction in MPI_AllReduce time (red)**
  - BICGSTAB requires $6s$ more MPI_AllReduce's than CA-BICGSTAB
  - Less than theoretical 24x since messages in CA-BICGSTAB are larger, not always latency-limited
- **P2P (blue) communication doubles** for CA-BICGSTAB
  - Basis computation requires twice as many SpMVs (P2P) per iteration as BICGSTAB





Representation of Matrix Values

Example: stencil with variable coefficients

implicit structure
explicit values

Example: general sparse matrix

explicit structure
explicit values

implicit structure
implicit values

Example: stencil with constant coefficients

explicit structure
implicit values

Example: Laplacian matrix of a graph

Representation of Matrix Structures

Hoemmen (2010), Fig 2.5