Efficient Deflation for Communication-Avoiding Krylov Subspace Methods

Erin Carson

Nicholas Knight, James Demmel

Univ. of California, Berkeley

Monday, June 24, NASCA 2013, Calais, France

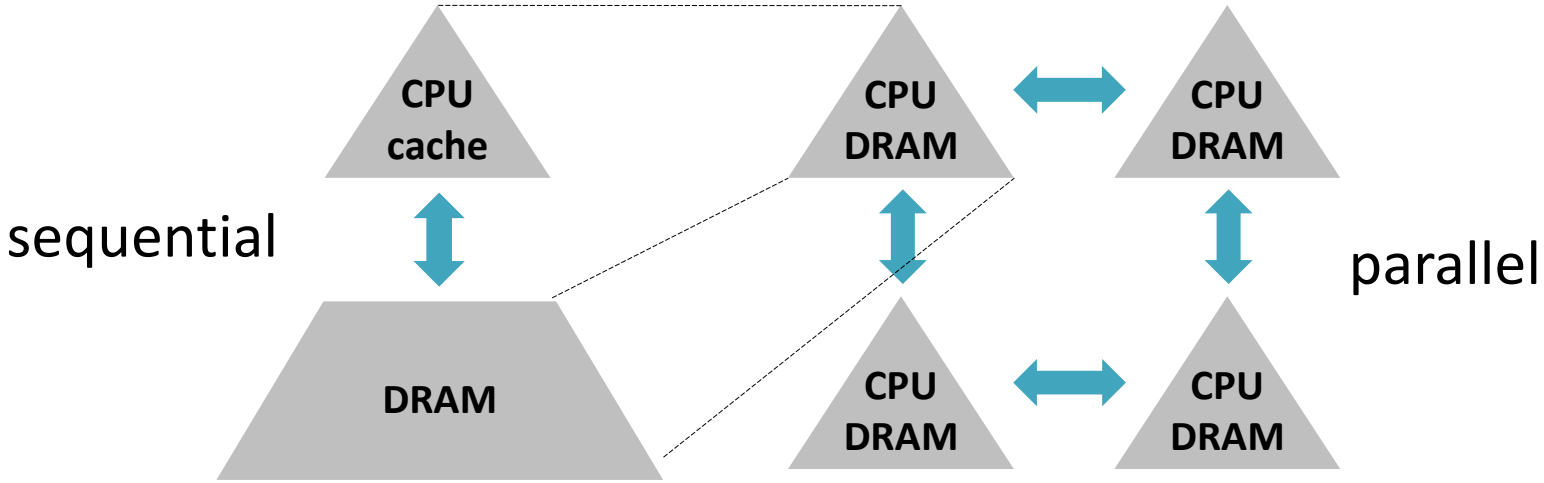
Overview

We derive the **Deflated Communication-Avoiding Conjugate Gradient** algorithm (Deflated CA-CG), demonstrating that deflation can be implemented while maintaining **asymptotic savings in data movement**.

1. Background
   - What is communication and why should it be avoided?
   - Communication-avoiding (s-step) conjugate gradient (CA-CG)
   - Deflated Conjugate Gradient method
2. Derivation of Deflated CA-CG
3. Asymptotic communication and computation costs
4. Evaluating tradeoffs in practice
   - Performance model and convergence results for model problem
5. Extensions and future work

What is Communication?

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



- On modern computers, communication is expensive, computation is cheap
  - Flop time << 1/bandwidth << latency
  - Communication a barrier to scalability (runtime and energy)
- **We must redesign algorithms to avoid communication**

How do Krylov Subspace Methods Work?

- A **Krylov subspace** is defined as

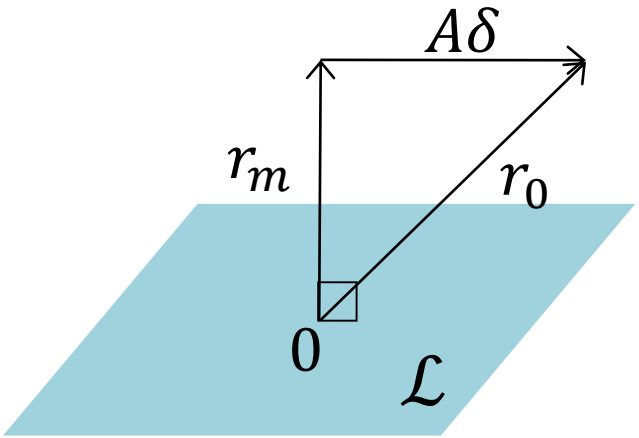
$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0\}$$

- A **Krylov subspace method (KSM)** is a projection process onto the subspace $\mathcal{K}$ orthogonal to $\mathcal{L}$
  - The choice of $\mathcal{L}$ distinguishes the various methods
    - Examples: Conjugate Gradient (CG), Generalized Minimum Residual Methods (GMRES), Biconjugate Gradient (BICG)

KSMs for solving linear systems: in iteration m, refine solution x_m to $Ax = b$ by imposing the condition

$$x_m = x_0 + \delta, \ \delta \in \mathcal{K}_m \quad \text{and} \quad r_0 - A\delta \perp \mathcal{L}_m,$$

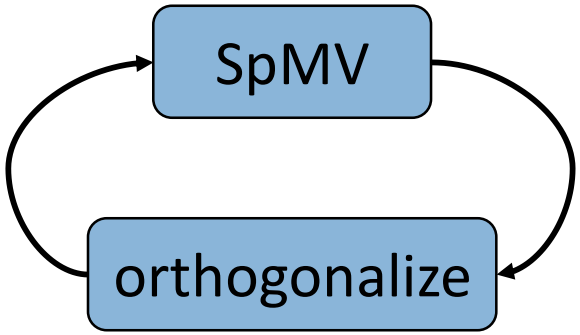
where $r_0 = b - Ax_0$



Communication Limits KSM Performance

In each iteration, the projection process proceeds by:

1. Adding a dimension to the Krylov subspace $\mathcal{K}_m$
   - Requires **sparse matrix-vector multiplication (SpMV)**
     - Parallel: communicate vector entries with neighbors
     - Sequential: read A (and N-vectors) from slow memory
2. Orthogonalizing with respect to $\mathcal{L}_m$
   - Requires inner products
     - Parallel: global reductions
     - Sequential: multiple reads/writes to slow memory



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

Classical Conjugate Gradient (CG)

Given: initial approximation x_0 for solving $Ax = b$

Let $p_0 = r_0 = b - Ax_0$

for $m = 0, 1, \ldots,$ until convergence **do**

$$\alpha_m = \frac{r_m^T r_m}{p_m^T A p_m}$$

$$x_{m+1} = x_m + \alpha_m p_m$$

$$r_{m+1} = r_m - \alpha_m A p_m$$

$$\beta_{m+1} = \frac{r_{m+1}^T r_{m+1}}{r_m^T r_m}$$

$$p_{m+1} = r_{m+1} + \beta_{m+1} p_m$$

end for

SpMVs and inner products require communication in each iteration!

CA-CG Derivation Overview

In iteration $m + s$ we have the relation

$$p_{m+s}, r_{m+s} \in \mathcal{K}_{s+1}(A, p_m) + \mathcal{K}_s(A, r_m)$$
$$x_{m+s} - x_m \in \mathcal{K}_s(A, p_m) + \mathcal{K}_{s-1}(A, r_m)$$

Let V be a basis for $\mathcal{K}_{s+1}(A, p_m) + \mathcal{K}_s(A, r_m)$, and let Gram matrix $\mathrm{G} = V^T V$.

For $1 \leq j \leq s$,

$$p_{m+j} = V p'_j \qquad r_{m+j} = V r'_j \qquad x_{m+j} - x_m = V x'_j$$

where $p'_j, r'_j,$ and x'_j are coordinates for $p_{m+j}, r_{m+j},$ and $x_{m+j} - x_m$ in basis V.

The product Ap_{m+j-1} can be written:

$$Ap_{m+j-1} = AVp'_{j-1} = VTp'_{j-1},$$

and inner products can be written:

$$r_{m+j}^T r_{m+j} = r_j'^T G r'_j \qquad p_{m+j-1}^T A p_{m+j-1} = p_{j-1}'^T G T p'_{j-1}$$

Communication-Avoiding CG

- This formulation allows an **$O(s)$ reduction in communication**
- Main idea: Split iteration loop into outer loop (k) and inner loop (j)

  Outer iteration: 1 communication step

  - Compute V_k: read A/communicate vectors only once (for well-partitioned A) using matrix powers kernel (see, e.g., Hoemmen et al., 2007)
  - Compute Gram matrix $G_k = V_k^T V_k$: one global reduction

  Inner iterations: s computation steps

  - Perform iterations $sk + j$, for $0 \leq j < s$, with **no communication**
  - Update $(2s + 1)$-vectors of coordinates of $p_{sk+j}, r_{sk+j}, x_{sk+j} - x_{sk}$ in V_k, replacing SpMVs and inner products
    - Quantities either local (parallel) or fit in fast memory (sequential)

Many CA-KSMs (or s-step KSMs) derived in the literature:
(Van Rosendale, 1983), (Walker, 1988), (Leland, 1989), (Chronopoulos and Gear, 1989), (Chronopoulos and Kim, 1990, 1992), (Chronopoulos, 1991), (Kim and Chronopoulos, 1991), (Joubert and Carey, 1992), (Bai, Hu, Reichel, 1991), (Erhel, 1995), (De Sturler, 1991), (De Sturler and Van der Vorst, 1995), (Toledo, 1995), (Chronopoulos and Kinkaid, 2001), (Hoemmen, 2010).

CA-Conjugate Gradient (CA-CG)

Given: initial approximation x_0 for solving $Ax = b$

Let $p_0 = r_0 = b - Ax_0$

for $k = 0, 1, \ldots,$ until convergence **do**

Calculate P_k, R_k, bases for $\mathcal{K}_{s+1}(A, p_{sk}), \mathcal{K}_s(A, r_{sk})$, resp.

Let $V_k = [P_k, R_k]$ and compute $G_k = V_k^T V_k$

Let $x'_0 = 0_{2s+1},\ r'_0 = [0^T_{s+1}, 1, 0^T_{s-1}]^T,\ p'_0 = [1, 0^T_{2s}]^T$

for $j = 0, \ldots, s-1$ **do**

$$\alpha_{sk+j} = \frac{r_j'^T G_k r'_j}{p_j'^T G_k T_k p'_j}$$

$$x'_{j+1} = x'_j + \alpha_{sk+j} p'_j$$

$$r'_{j+1} = r'_j - \alpha_{sk+j} T_k p'_j$$

$$\beta_{sk+j+1} = \frac{r_{j+1}'^T G_k r'_{j+1}}{r_j'^T G_k r'_j}$$

$$p'_{j+1} = r'_{j+1} + \beta_{sk+j+1} p'_j$$

end for

Compute $x_{sk+s} = V_k x'_s + x_{sk},\ r_{sk+s} = V_k r'_s, p_{sk+s} = V_k p'_s$

end for

via CA Matrix Powers Kernel

Global reduction to compute G

Local computations within inner loop require no communication!

Deflated CG (Saad et al., 2000)

- Deflation: removing eigenvalues that are hard to converge to in order to increase convergence rate
- Convergence of CG governed by $\boldsymbol{\kappa(A) = \lambda_N/\lambda_1}$
  - Where $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ are eigenvalues of A
- Let W be an $N \times c$ matrix to be used in deflation
- Deflated CG is equivalent to CG with system $H^T A H \tilde{x} = H^T b$ where $H = I - W(W^T AW)^{-1}(AW)^T$ is the matrix of the A-orthogonal projection onto $W^{\perp_A}$
  - When columns of W are approximate eigenvectors of A associated with $\lambda_1, \lambda_2, \ldots, \lambda_c$, $\boldsymbol{\kappa\left(H^T AH\right) \approx \lambda_N/\lambda_{c+1}}$
    - Deflated CG should increase rate of convergence

Can deflation techniques be applied to CA-CG while maintaining asymptotic reduction in communication cost?

Deflated CG Algorithm (Saad et al., 2000)

Define W to be a length $N \times c$ basis. Compute $W^T AW$.

Compute $x_0 = W(W^T AW)^{-1} W^T b$

$r_0 = b - Ax_0, \mu_0 = (W^T AW)^{-1} W^T Ar_0, p_0 = r_0 - W\mu_0$

for $m = 0, 1, \ldots,$ until convergence **do**

$\alpha_m = r_m^T r_m / p_m^T Ap_m$

$x_{m+1} = x_m + \alpha_m p_m$

$r_{m+1} = r_m - \alpha_m Ap_m$

$\beta_{m+1} = r_{m+1}^T r_{m+1} / r_m^T r_m$

Solve $W^T AW \mu_{m+1} = W^T Ar_{m+1}$ for μ_{m+1}

$p_{m+1} = \beta_{m+1} p_m + r_{m+1} - W\mu_{m+1}$

end for

SpMVs and dot products required in each inner loop, as in CG

New term due to deflation; requires SpMV and global reduction

Avoiding Communication in Deflation Process

In Deflated CG, we have

$$p_{sk+j}, r_{sk+j} \in \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk}) + \mathcal{K}_{s-1}(A, W)$$
$$x_{sk+j} - x_{sk} \in \mathcal{K}_s(A, p_{sk}) + \mathcal{K}_{s-1}(A, r_{sk}) + \mathcal{K}_{s-2}(A, W)$$

To compute μ_{sk+j+1}, we also need

$$Ar_{sk+j+1} \in \mathcal{K}_{s+2}(A, p_{sk}) + \mathcal{K}_{s+1}(A, r_{sk}) + \mathcal{K}_s(A, W)$$

Let V_k be an $N \times (2s + 3 + cs)$ matrix whose columns span this space, i.e.,

$$V_k \in \mathcal{K}_{s+2}(A, p_{sk}) + \mathcal{K}_{s+1}(A, r_{sk}) + \mathcal{K}_s(A, W).$$

If we compute $G_k = V_k^T V_k$, and extract $Z_k = W^T V_k$ from G_k, then

$$W^T A r_{sk+j+1} = Z_k T_k r'_{k,j+1}.$$

As in CA-CG, we compute inner products and mult. by A in the inner loop by updating length-$(2s + 3 + cs)$ coordinate vectors in basis V_k.

Deflated CA-CG

Define W to be a length $N \times c$ basis. Compute $W^T AW$.

$x_0 = W(W^T AW)^{-1} W^T b, r_0 = b - Ax_0, \mu_0 = (W^T AW)^{-1} W^T Ar_0, p_0 = r_0 - W\mu_0$

Compute $\mathcal{W}$, a basis for $\mathcal{K}_s(A, W)$

for $k = 0, 1, \ldots,$ until convergence **do**

Compute P_k, R_k, bases for $\mathcal{K}_{s+2}(A, p_{sk}), \mathcal{K}_{s+1}(A, r_{sk})$, resp.

Construct T_k such that $A\left[\underline{P_k}, \underline{R_k}, \underline{\mathcal{W}}\right] = [P_k, R_k, \mathcal{W}]T_k$

Let $V_k = [P_k, R_k, \mathcal{W}]$, compute $G_k = V_k^T V_k, Z_k = W^T V_k$

$p'_0 = [1, 0^T_{2s+2+cs}]^T, r'_0 = [0^T_{s+2}, 1, 0^T_{s+cs}]^T, x'_0 = 0_{2s+3+cs}$

for $j = 0$ to $s - 1$ **do**

$\alpha_{sk+j} = r_j'^T G_k r'_j / p'^T_j G_k T_k p'_j$

$x'_{j+1} = x'_j + \alpha_{sk+j} p'_j$

$r'_{j+1} = r'_j - \alpha_{sk+j} T_k p'_j$

$\beta_{sk+j+1} = r'^T_{j+1} G_k r'_{j+1} / r'^T_j G_k r'_j$

Solve $W^T AW \mu_{sk+j+1} = Z_k T_k r'_j$ for μ_{sk+j+1}

$p'_{j+1} = \beta_{sk+j+1} p'_j + r'_{j+1} - [0^T_{2s+3}, \mu^T_{sk+j+1}, 0^T_{c(s-1)}]^T$

end for

$x_{sk+s} = V_k x'_s + x_{sk}, r_{sk+s} = V_k r'_s, p_{sk+s} = V_k p'_s$

end for

One-time (offline) call to CA matrix powers kernel with c vectors

Additional bandwidth cost once per s iterations

Local operations, requires no communication

Computation and Communication Complexity

Model Problem (2D Laplacian), s iterations of parallel algorithm

	Flops	Words moved	Messages
CG	$O\left(\frac{sN}{p}\right) + O(s)$	$O\left(s\sqrt{N/p}\right) + O(s)$	$O(s \log_2 p) + O(s)$
CA-CG	$O\left(\frac{s^2N}{p}\right) + O(s^3)$	$O\left(s\sqrt{N/p}\right) + O(s^2)$	$O(\log_2 p)$
Deflated CG	$O\left(\frac{csN}{p}\right) + O(c^2s)$	$O\left(s\sqrt{N/p}\right) + O(cs)$	$O(s \log_2 p) + O(s)$
Deflated CA-CG	$O\left(\frac{cs^2N}{p}\right) + O(c^2s^3)$	$O\left(s\sqrt{N/p}\right) + O(cs^2)$	$O(\log_2 p)$

Note: offline costs of computing and factoring W^TAW omitted for Deflated CG and Deflated CA-CG (as well as computing $\mathcal{K}_s(A, W)$ for Deflated CA-CG)

Is This Efficient in Practice?

- In practice, **evaluating tradeoffs between s and c is nontrivial**
  - Larger s means faster speed per iteration, but can potentially decrease convergence rate in finite precision
  - Larger c gives better theoretical convergence rate, but can potentially decrease speed per iteration

- Performance modeling for a specific problem, method, and machine must take both
  1. How **time per iteration** changes with s and c
  2. How the **number of iterations required for convergence** changes with s and c, and

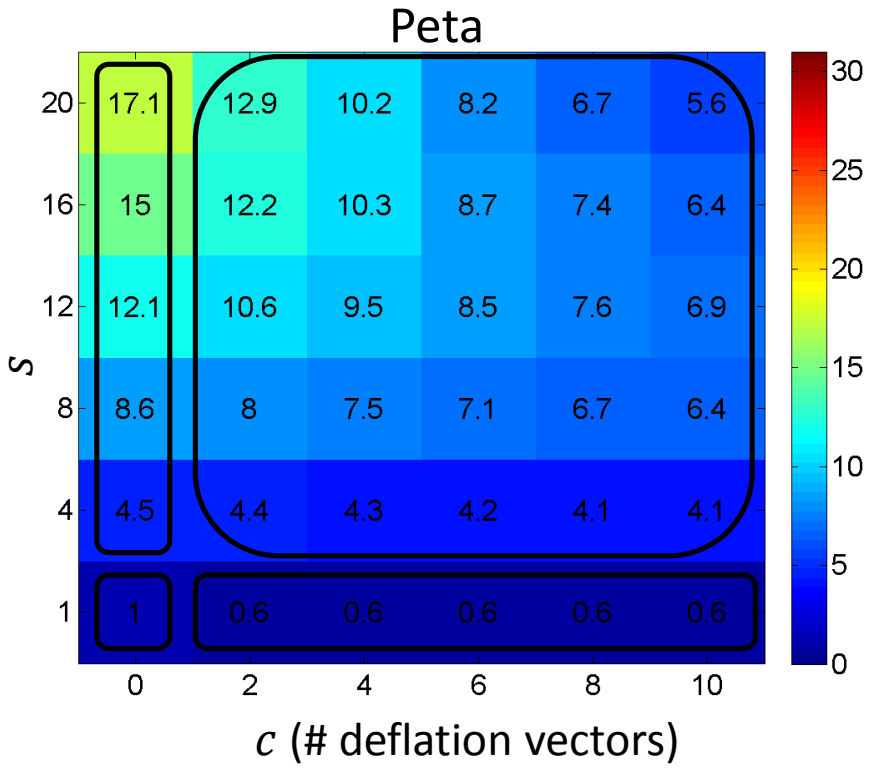
  into account.

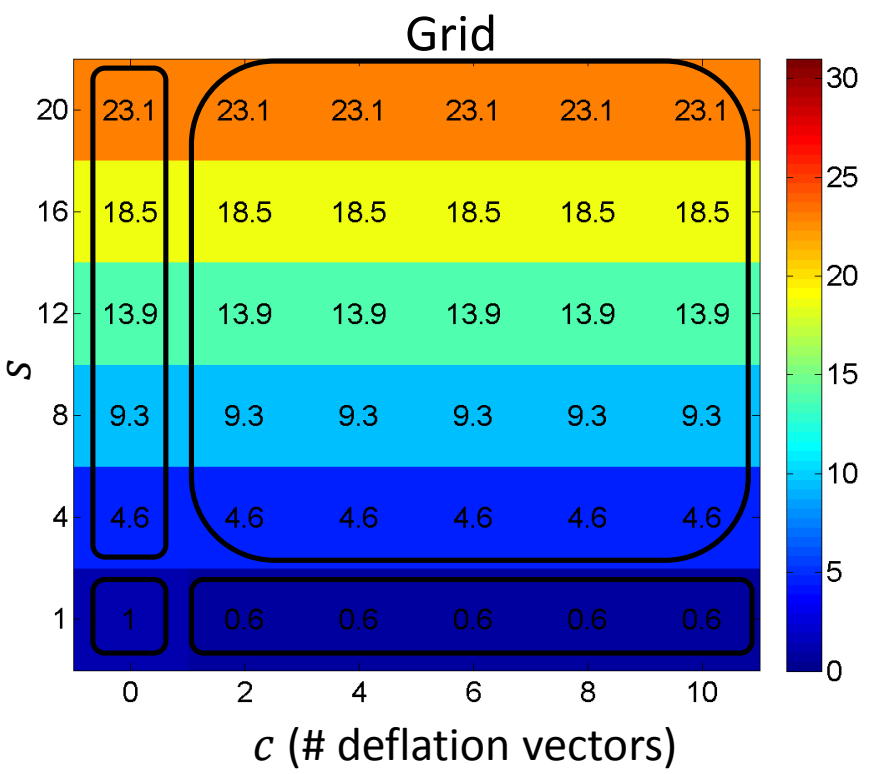
- We will demonstrate the tradeoffs involved for our model problem (2D Laplacian) on two large distributed memory machine models

CA Speedup per Iteration

Plot of modeled speedup per iteration relative to CG for 2 machines, for 2D Laplacian with $N = 262{,}144$, $p = 512$ where

Time $= \gamma$(arithmetic operations) $+ \beta$(words moved) $+ \alpha$(messages sent)

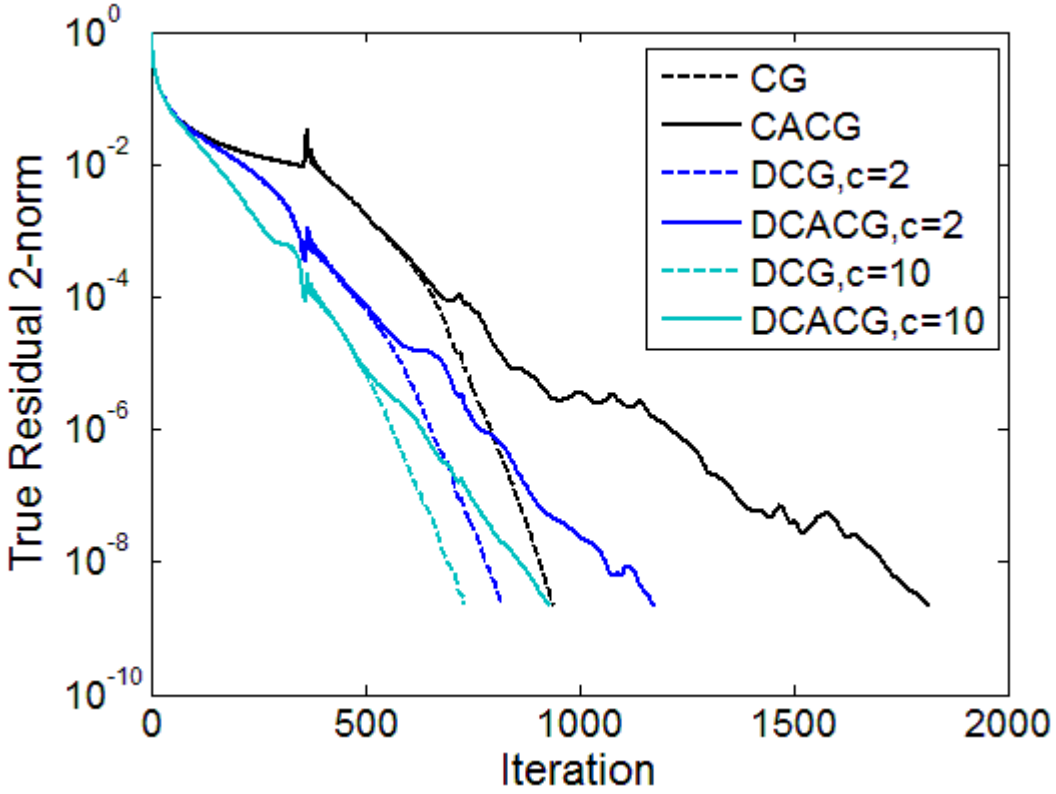




Peta: $\gamma = 2 \cdot 10^{-11}$(s/flop), $\alpha = 10^{-5}$(s), $\beta = 2 \cdot 10^{-9}$(s/word)

Grid: $\gamma = 10^{-12}$(s/flop), $\alpha = 10^{-1}$(s), $\beta = 25 \cdot 10^{-9}$(s/word)

Convergence for Model Problem



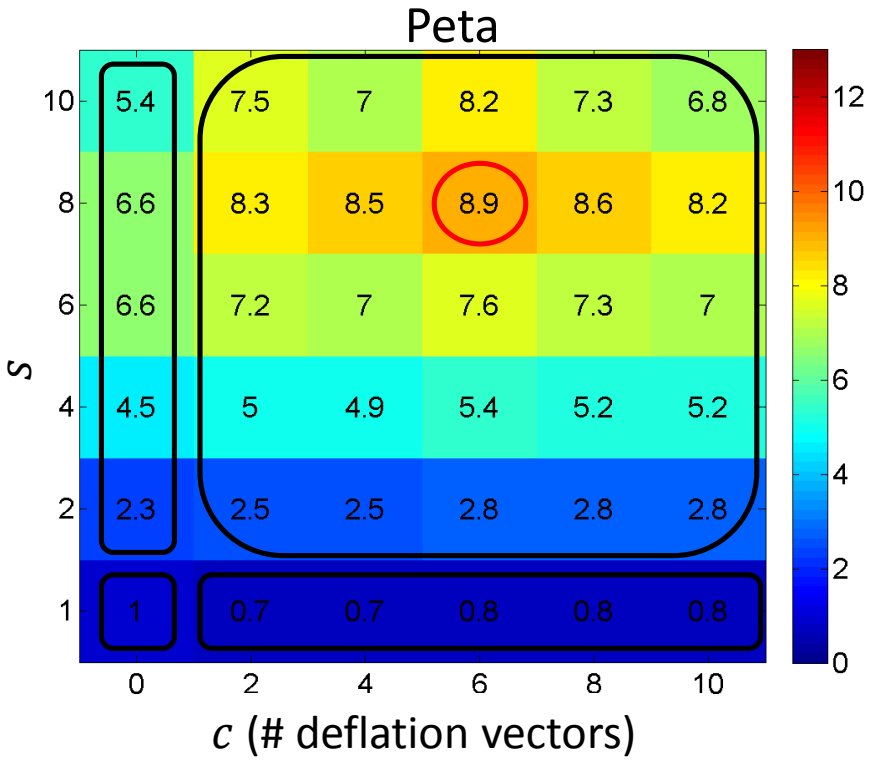
Monomial Basis,
$s = 10$

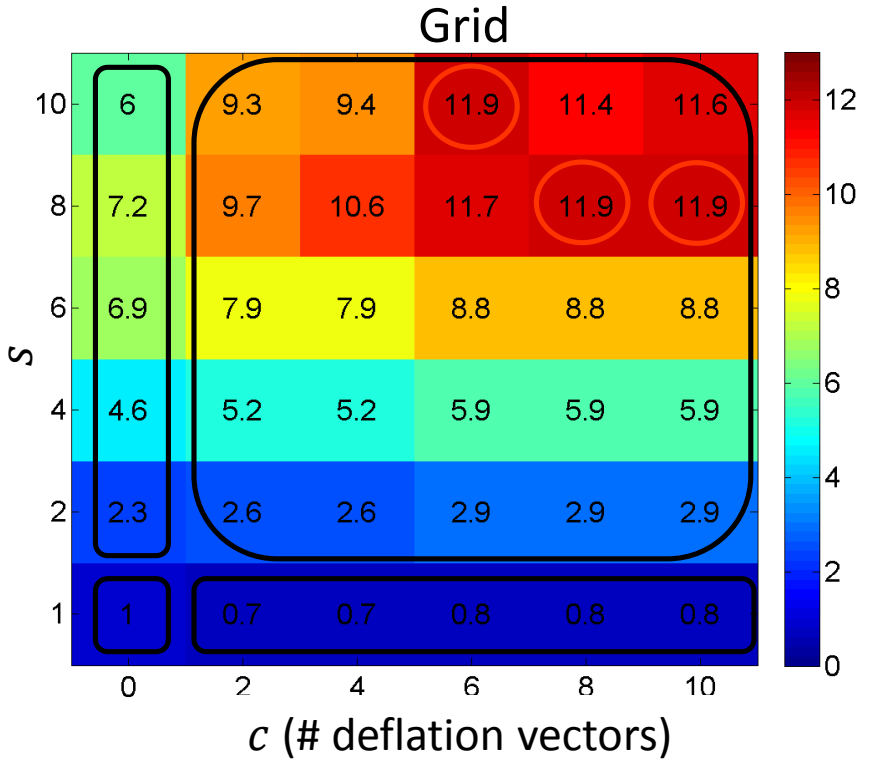
$\rho_0(A) = 1,$
$\rho_j(A) = A \cdot \rho_{j-1}(A)$

Matrix: 2D Laplacian(512), $N = 262{,}144$. Right hand side set such that true solution has entries $x_i = 1/\sqrt{n}$. Deflated CG algorithm (DCG) from (Saad et al., 2000).

Total Speedup, Monomial Basis

Total speedup = (speedup per iteration) × (number of iterations(Monomial))

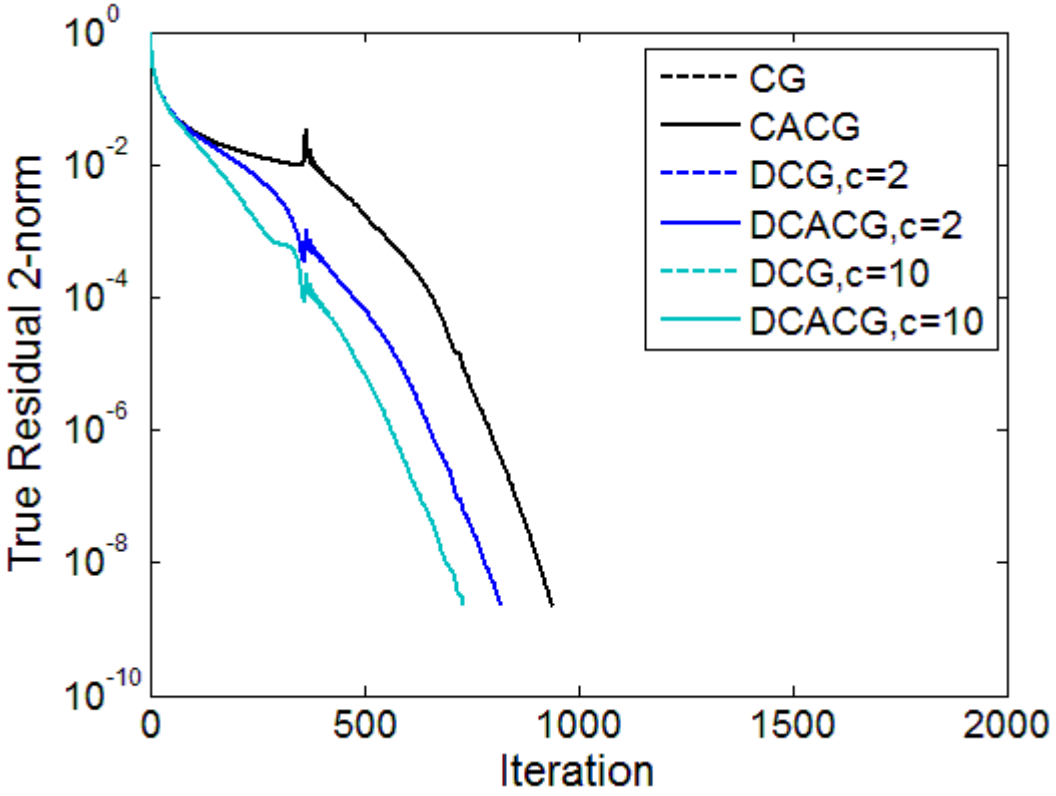




- Since CA-CG method suffers delayed convergence with monomial basis, higher s doesn't always give better performance (convergence fails for $s > 10$).
- On Peta, since relative latency is not as bad as on Grid, speedups decrease for large c values.

Convergence for Model Problem

A better choice of basis leads to stability for higher s values:



Newton Basis,
$s = 20$

$\rho_0(A) = 1,$
$\rho_j(A) = (A - \theta_j I)\rho_{j-1}(A)$

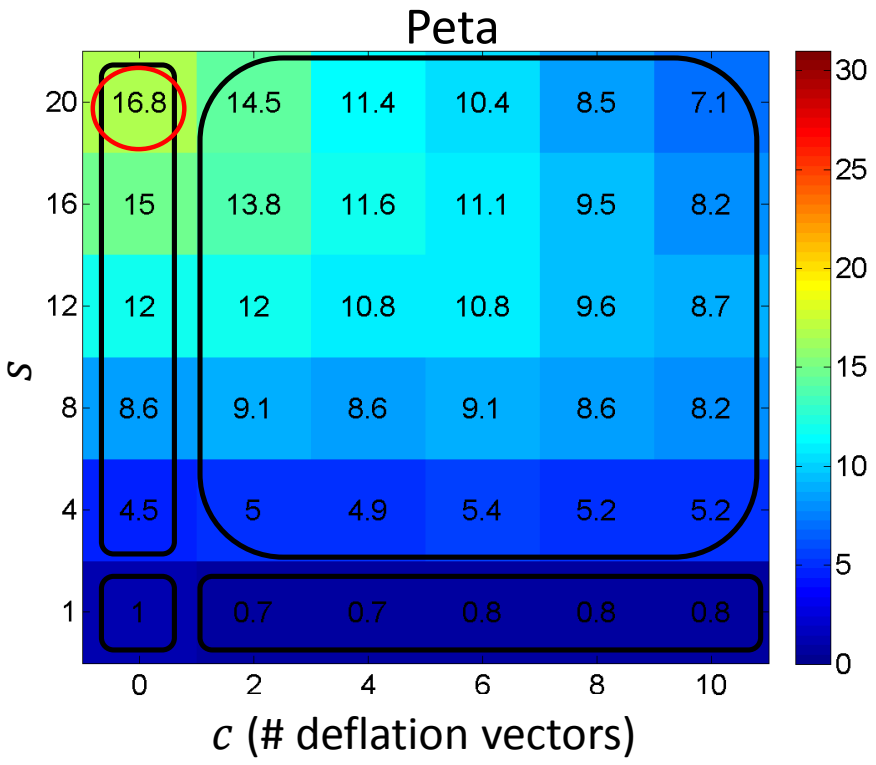
where θ_j are Leja-ordered points on $\mathcal{F}(A)$

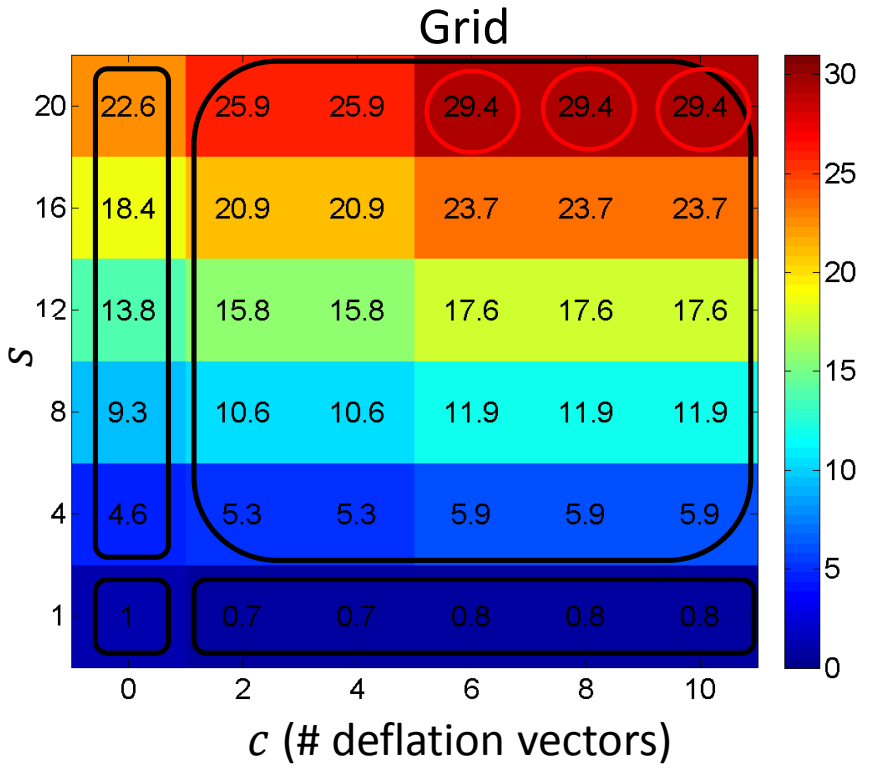
*For details on better bases for Krylov subspaces, see, e.g., Phillipe and Reichel, 2012.

Matrix: 2D Laplacian(512), N $= 262{,}144$. Right hand side set such that true solution has entries $x_i = 1/\sqrt{n}$. Deflated CG algorithm (DCG) from (Saad, et al., 2000).

Total Speedup

Total speedup = (speedup per iteration) × (number of iterations(Newton))





- Peta: Speedup decreases with increasing c; CA deflation doesn't lead to significant overall performance improvements over CA-CG
- Grid: since O(s) speedup from CA techniques remains constant for increasing c, CA deflation increases overall speedup for all s values!

Conclusions and Future Work

- Summary
  - **Proof-of-concept that deflation techniques can be implemented in CA-CG in a way that still avoids communication**
    - Asymptotic bounds confirm that Deflated CA-CG maintains the $O(s)$ reduction in latency over CG and Deflated CG
  - Performance modeling demonstrates **nontrivial tradeoffs between speed per iteration and convergence rate** for different methods
- Future Work
  - Extending other deflation techniques to CA methods
    - Solving (slowly-changing) series of linear systems (recycling Krylov subspaces)
    - Reorthogonalization to fix instability in Deflated CA-CG ($r \perp W$ fails in finite precision, set $r = (I - W(W^T W)^{-1} W^T) r$) (Saad et al., 2000)
    - Equivalent 'augmented' formulations (Gaul et al., 2013)
  - Claim: CA deflation can be applied to other deflated Krylov methods
    - GMRES, MINRES, BICG(STAB), QMR, Arnoldi, Lanczos, etc., see, e.g., (Gutknecht, 2012)

Thank you!

Erin Carson: ecc2z@cs.berkeley.edu
Nick Knight: knight@cs.berkeley.edu

Extra Slides

CA-CG Derivation Overview

In iteration $sk + j$, for $s > 0$, $0 \leq j \leq s$, we exploit the relation

$$p_{sk+j}, r_{sk+j} \in \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$$
$$x_{sk+j} - x_{sk} \in \mathcal{K}_s(A, p_{sk}) + \mathcal{K}_{s-1}(A, r_{sk})$$

Let V_k be a matrix whose columns span $\mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$.

Then for iterations $sk + 1$ through $sk + s$, we can implicitly update the length n vectors p_{sk+j}, r_{sk+j}, and $x_{sk+j} - x_{sk}$ by updating their coordinates (length $2s + 1$ vectors) in basis V_k.

$$p_{sk+j} = V_k p'_{k,j} \qquad r_{sk+j} = V_k r'_{k,j} \qquad x_{sk+j} - x_{sk} = V_k x'_{k,j}$$

CA-CG Derivation Overview

In iteration $sk + j$, for $s > 0$, $0 \leq j \leq s$, we exploit the relation

$$p_{sk+j}, r_{sk+j} \in \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$$
$$x_{sk+j} - x_{sk} \in \mathcal{K}_s(A, p_{sk}) + \mathcal{K}_{s-1}(A, r_{sk})$$

If we compute basis V_k for $\mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$ and compute Gram matrix $G_k = V_k^T V_k$, then for iterations $0 \leq j < s$ we can implicitly update the length n vectors p_{sk+j}, r_{sk+j}, and $x_{sk+j} - x_{sk}$ by updating their coordinates (length $2s + 1$ vectors) in basis V_k.

$$p_{sk+j} = V_k p'_{k,j} \qquad r_{sk+j} = V_k r'_{k,j} \qquad x_{sk+j} - x_{sk} = V_k x'_{k,j}$$

The product Ap_{sk+j} can be represented implicitly in basis V_k by $B_k p'_{k,j}$, since

$$Ap_{sk+j} = AV_k p'_{k,j} = V_k B_k p'_{k,j},$$

and we can write dot products as

$$r_{sk+j}^T r_{sk+j} = r_{k,j}'^T G_k r'_{k,j} \qquad p_{sk+j}^T A p_{sk+j} = p_{k,j}'^T G_k B_k p'_{k,j}$$

CA-CG Derivation Overview

The product Ap_{sk+j} can be represented implicitly in basis V_k by $B_k p'_{k,j}$, since

$$Ap_{sk+j} = AV_k p'_{k,j} = V_k B_k p'_{k,j}$$

If we compute $G_k = V_k^T V_k$, we can write dot products as

$$r_{sk+j}^T r_{sk+j} = r_{k,j}'^T G_k r'_{k,j} \qquad p_{sk+j}^T A p_{sk+j} = p_{k,j}'^T G_k B_k p'_{k,j}$$

B_k and G_k are small $O(s) \times O(s)$ matrices that fits in fast/local memory; Multiplication by B_k and G_k require no communication!

Related Work: s-step methods

Authors	KSM	Basis	Precond?	Mtx Pwrs?	TSQR?
Van Rosendale, 1983	CG	Monomial	Polynomial	No	No
Leland, 1989	CG	Monomial	Polynomial	No	No
Walker, 1988	GMRES	Monomial	None	No	No
Chronopoulos and Gear, 1989	CG	Monomial	None	No	No
Chronopoulos and Kim, 1990	Orthomin, GMRES	Monomial	None	No	No
Chronopoulos, 1991	MINRES	Monomial	None	No	No
Kim and Chronopoulos, 1991	Symm. Lanczos, Arnoldi	Monomial	None	No	No
de Sturler, 1991	GMRES	Chebyshev	None	No	No

Related Work: s-step methods

Authors	KSM	Basis	Precond?	Mtx Pwrs?	TSQR?
Joubert and Carey, 1992	GMRES	Chebyshev	No	Yes*	No
Chronopoulos and Kim, 1992	Nonsymm. Lanczos	Monomial	No	No	No
Bai, Hu, and Reichel, 1991	GMRES	Newton	No	No	No
Erhel, 1995	GMRES	Newton	No	No	No
de Sturler and van der Vorst, 1995	GMRES	Chebyshev	General	No	No
Toledo, 1995	CG	Monomial	Polynomial	Yes*	No
Chronopoulos and Swanson, 1996	CGR, Orthomin	Monomial	No	No	No
Chronopoulos and Kinkaid, 2001	Orthodir	Monomial	No	No	No

Convergence in Finite Precision

- CA-KSMs are mathematically equivalent to classical KSMs

- But have different behavior in finite precision!
  - Roundoff error causes **delay of convergence**
  - Bounds on magnitude of roundoff error increase with s

- In solving practical problems, roundoff error can **limit performance**
  - If # iterations increases more than time per iteration decreases due to CA techniques, **no speedup expected!**

- To perform a practical performance comparison amongst CG, Deflated CG, CA-CG and Deflated CA-CG, **we must combine speedup per iteration with the total number of iterations for each method**

Detailed Complexity Analysis: CG

$$Flops_{CG} = 2s - \frac{2s}{p} + \frac{19ns}{p}$$

$$Words_{CG} = 2s - \frac{2s}{p} + 4s\sqrt{n/p}$$

$$Mess_{CG} = 4s + 2slog_2p$$

Detailed Complexity Analysis: CA-CG

$$Flops_{CACG} = 49s - 2s^2/p + 36s^2\sqrt{n/p} + 25n/p - 3s/p + 72s\sqrt{n/p} - 1/p + 74s^2 + 36s^3 + 36\sqrt{n/p} + 36ns/p + 4ns^2/p + 12$$

$$Words_{CACG} = 11s - 2s^2/p - 3s/p + 8s\sqrt{n/p} - 1/p + 6s^2 + 8\sqrt{n/p} + 5$$

$$Mess_{CACG} = \log_2 p + 8$$

Detailed Complexity Analysis: DCG

$$Flops_{DCG} = 2s + 2c^2s - 2s/p - cs/p + 29ns/p + 4cns)/p$$

$$Words_{DCG} = 2s + cs - 2s/p + 8s\sqrt{n/p} - cs/p$$

$$Mess_{DCG} = 8s + 3slog_2p$$

Detailed Complexity Analysis: DCACG

$$Flops_{DCACG} = 240s - 2s^2/p + 36s^2\sqrt{n/p} + 5cs + 60cs^2 + 2c^2s + 32cs^3 + 65n/p - 7s/p + 144s\sqrt{n/p} - 6/p + 184s^2 + 44s^3 + 2c^2s^2 + 8c^2s^3 + 144\sqrt{n/p} - 3cs/p + 44ns/p - 2cs^2/p + 4ns^2/p + 12cns/p + 4cns^2/p + 96$$

$$Words_{DCACG} = 23s - 2s^2/p + 3cs + 2cs^2 - 7s/p + 8s\sqrt{n/p} - 6/p + 6s^2 + 16\sqrt{n/p} - 3cs/p - 2cs^2/p + 22$$

$$Mess_{DCACG} = \log_2 p + 8$$

Better Polynomial Bases

In general, columns v_{i+1} of V computed by the 3-term recurrence

$$v_{i+1} = \left((A - \hat{\alpha}_i I)v_i - \hat{\beta}_i v_{i-1}\right)/\hat{\gamma}_i$$

Scaled Monomial: For scalars $\{\sigma_i\}_{i=1}^{s}$,

$$\hat{\alpha}_i = 0, \hat{\beta}_i = 0, \hat{\gamma}_i = \sigma_i$$

Newton: For Leja-ordered Ritz values $\{\theta_i\}_{i=1}^{s}$ and scalars $\{\sigma_i\}_{i=1}^{s}$

$$\hat{\alpha}_i = \theta_i, \hat{\beta}_i = 0, \hat{\gamma}_i = \sigma_i$$

Chebyshev: Given bounding ellipse for spectrum with foci at $d \pm c$, the scaled and shifted polynomials are

$$\tilde{\tau}_i(z) = \tau_i((d - z)/c)/\sigma_i,$$

$$\hat{\alpha}_i = d, \qquad \hat{\beta}_i = -\frac{c\sigma_i}{2\sigma_{i+1}}, \qquad \hat{\gamma}_i = -\frac{c\sigma_{i+1}}{2\sigma_i}$$

Leja Ordering

Let $\mathbb{S}$ be a compact set in $\mathbb{C}$. (Note that here $\mathbb{S}$ is set of approx. Ritz values). Then a Leja ordering can be computed by

$$\theta_1 = \text{argmax}_{z \in \mathbb{S}} |z|$$

$$\theta_{i+1} = \text{argmax}_{z \in \mathbb{S}} \prod_{k=0}^{i} |z - z_k|$$

Leja (1957)

Many references for using different polynomials for Krylov subspace calculation:

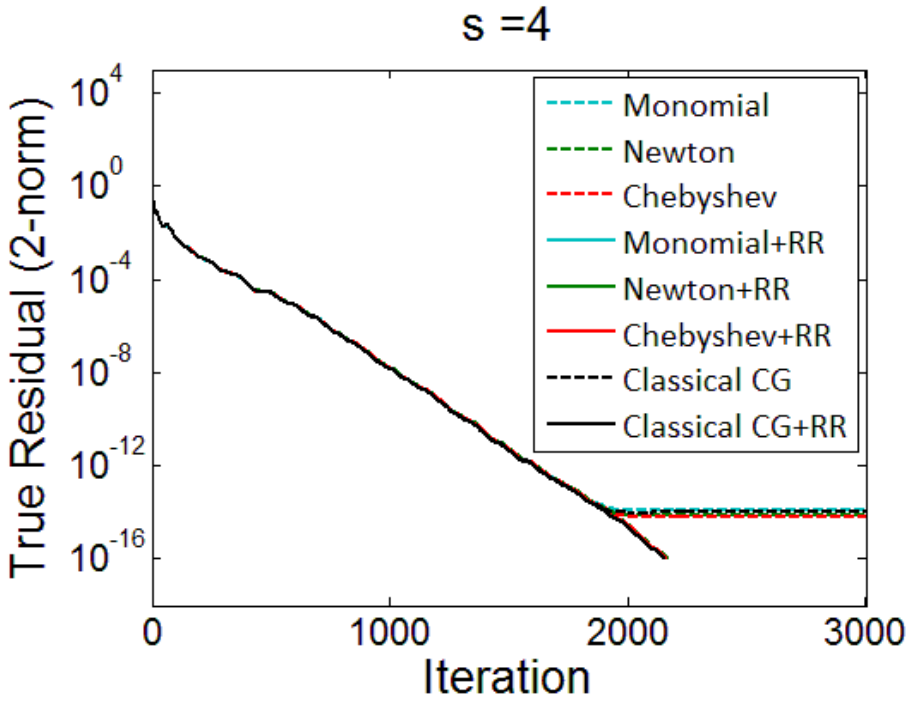
See, e.g., Philippe and Reichel (2012), Bai et al. (1994), Joubert and Carey (1992), de Sturler and van der Vorst (1994, 1995), Erhel (1995).

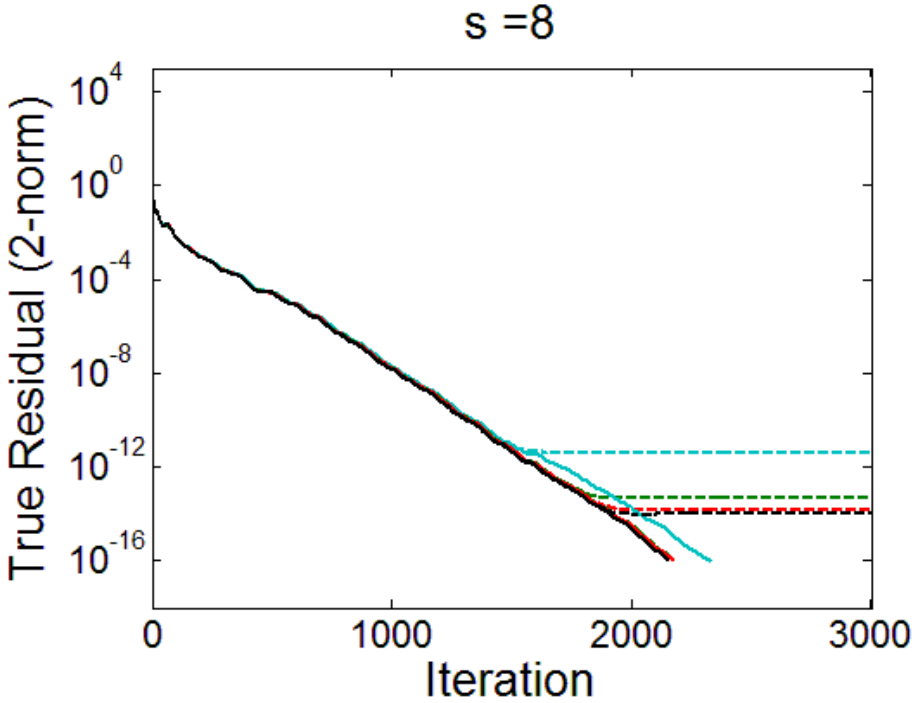
Residual Replacement Strategy

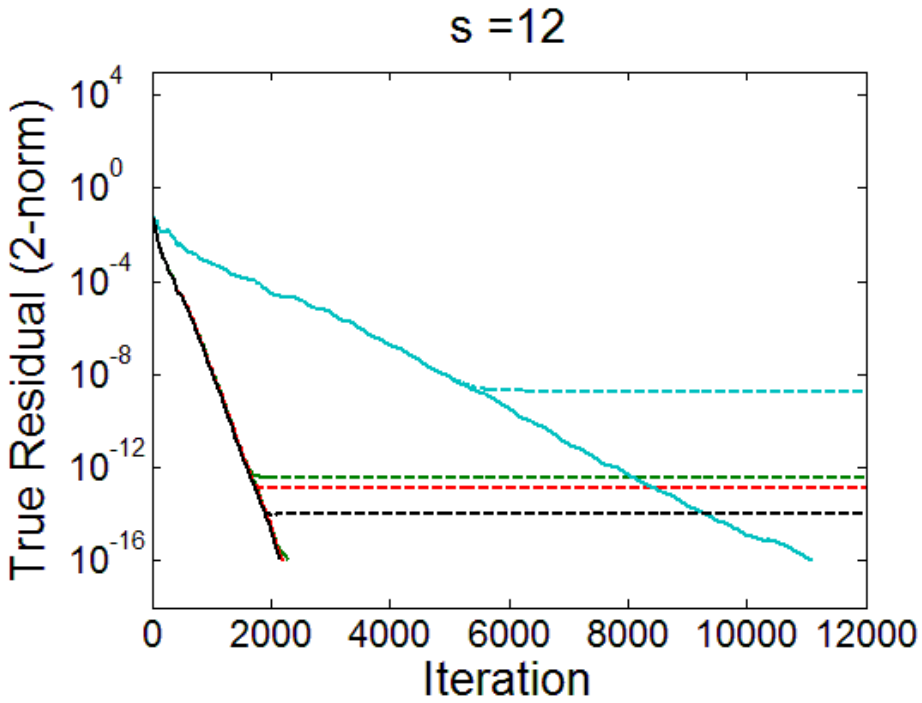
- Van der Vorst and Ye (1999) : Residual replacement used in combination with group-update to improve the maximum attainable accuracy
  - Given computable upper bound for deviation of residuals, replacement steps chosen to satisfy two constraints:
    1. Deviation must not grow so large that attainable accuracy is limited
    2. Replacement must not perturb Lanczos recurrence relation for computed residuals such that convergence deteriorates

- When the computed residual converges to level $O(\varepsilon)\|A\|\|x\|$ strategy reduces true residual, to level $O(\varepsilon)\|A\|\|x\|$

- We devise an analogous strategy for CA-CG and CA-BICG
  - Our strategy does not asymptotically increase communication or computation!



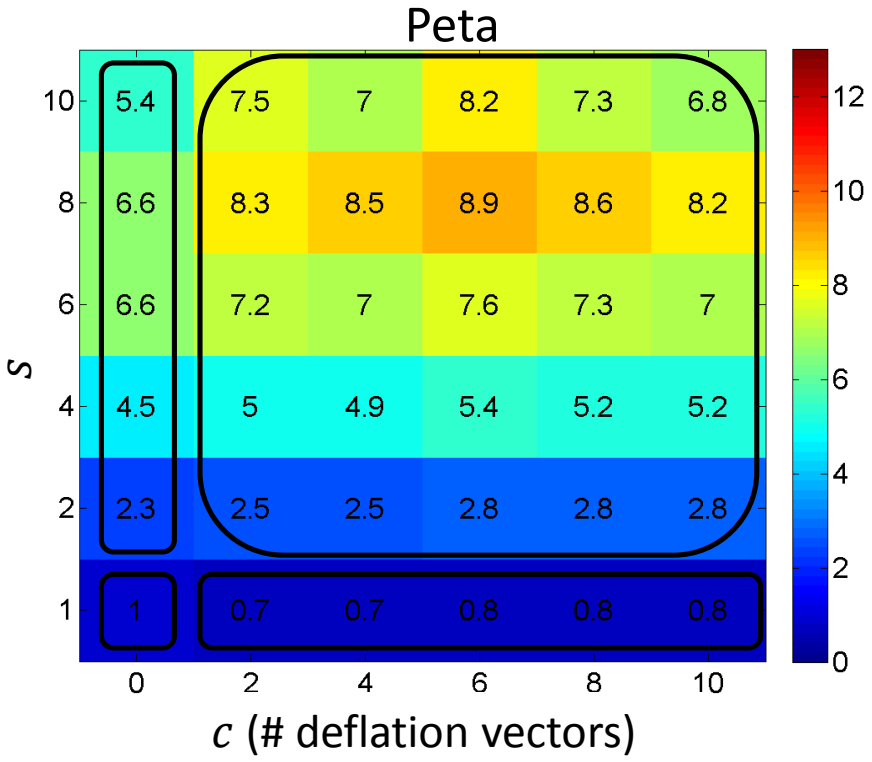


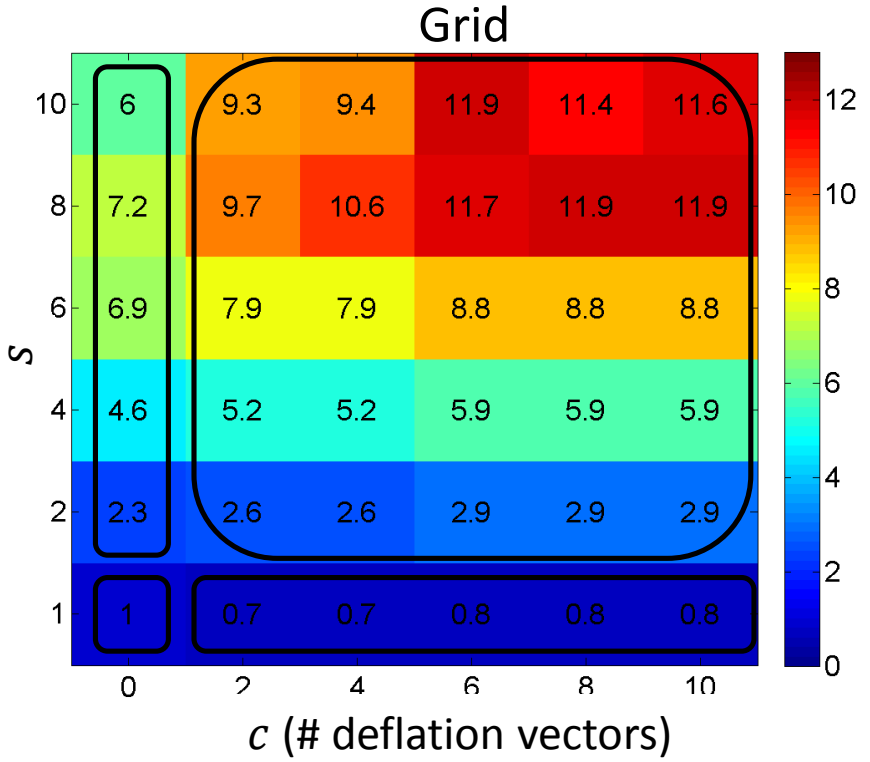


- Matrix: consph (FEM), SPD, $N = 8.3E4,\ NZ = 6E6, \kappa(A) \approx 9.7E3.$
- In all tests #replacements ≤ 5.
- **Orders of magnitude improvement in accuracy** for little additional cost!
- But doesn't fix slow convergence due to ill-conditioned basis.

Total Speedup, Monomial Basis

Total speedup = (speedup per iteration) × (number of iterations(Monomial))



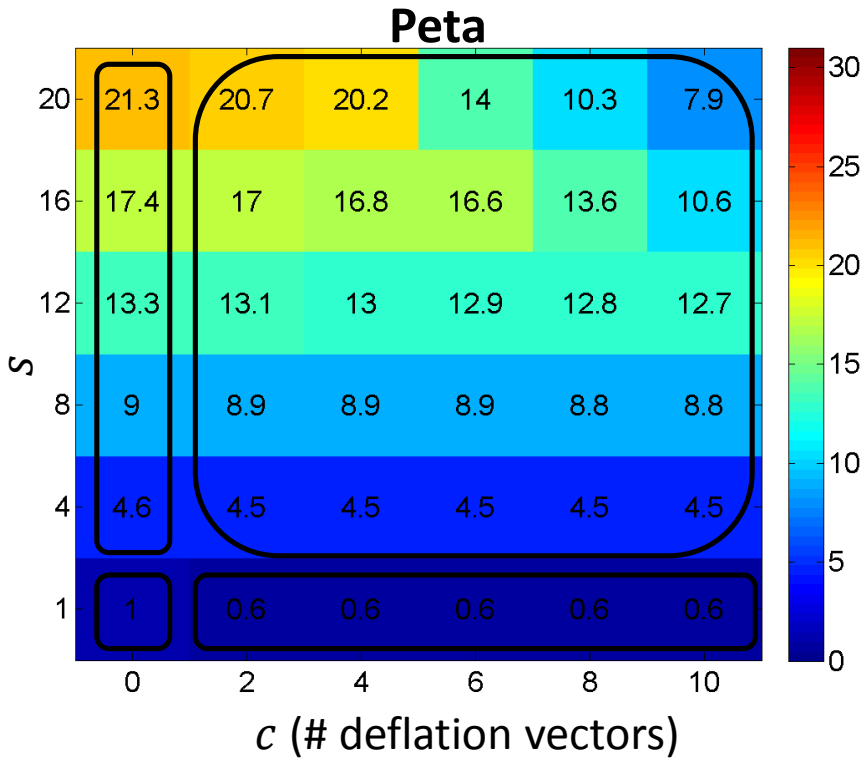


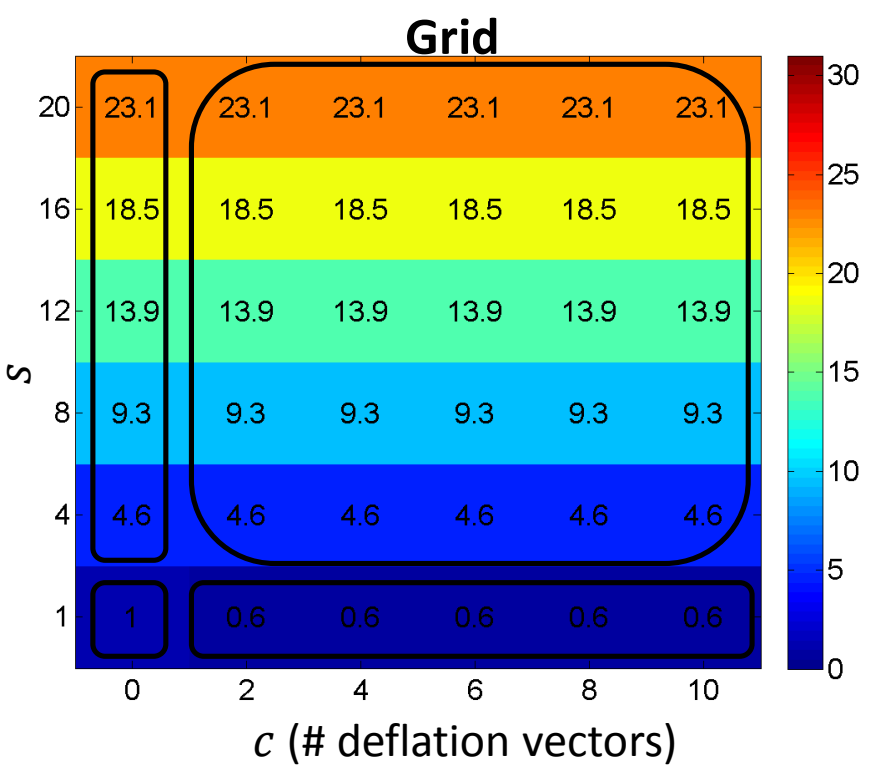
Since the CA-CG method without deflation suffers delayed convergence, deflation results in performance improvements on both machines (note that delayed convergence also means that higher s doesn't always give better performance for monomial). On Peta, since relative latency is not as bad as on Grid, speedups start to decrease for large c values.

Overlapping Communication and Computation

Plot of modeled speedup per iteration relative to CG for 2 machines, for 2D Laplacian with $n = 262{,}144$, $p = 512$ where

Time $= \max(\ \gamma(\text{arithmetic operations}),\ \beta(\text{words moved}) + \alpha(\text{messages sent}))$



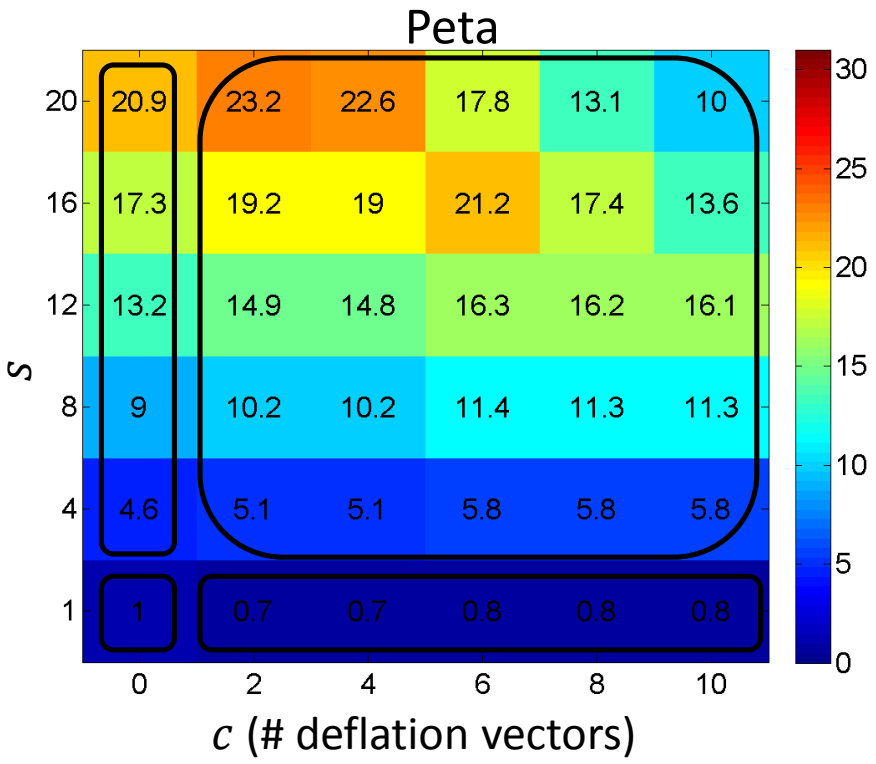


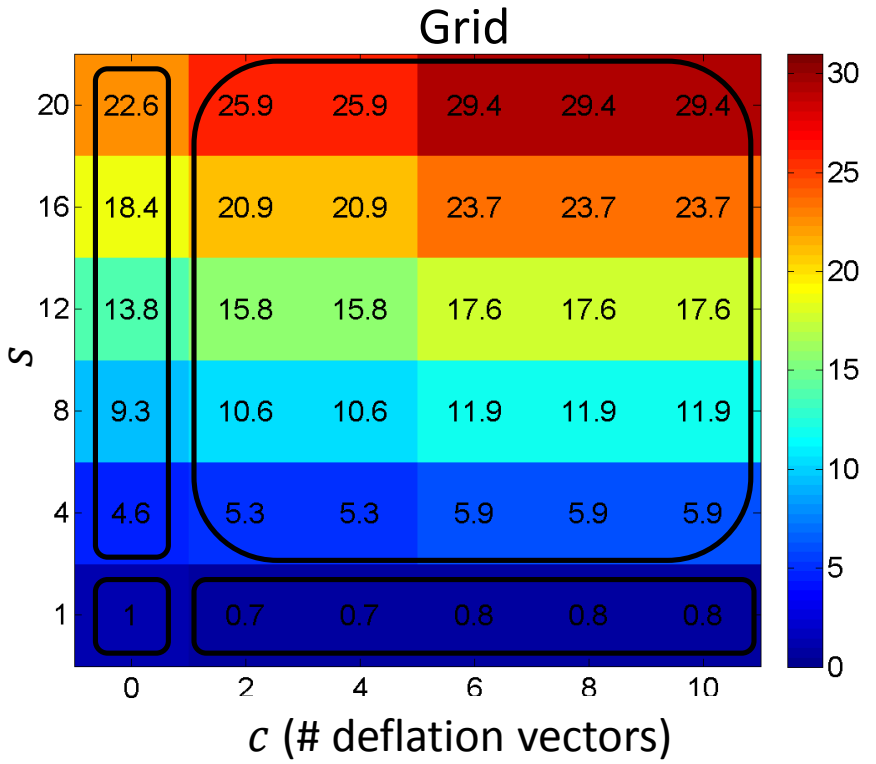
Peta: $\gamma = 2 \cdot 10^{-11}$(s/flop), $\alpha = 10^{-5}$(s), $\beta = 2 \cdot 10^{-9}$(words/s)

Grid: $\gamma = 10^{-12}$(s/flop), $\alpha = 10^{-1}$(s), $\beta = 25 \cdot 10^{-9}$(words/s)

Total Speedup with Overlap (Newton)

Total speedup = (speedup per iteration) × (number of iterations(Newton))



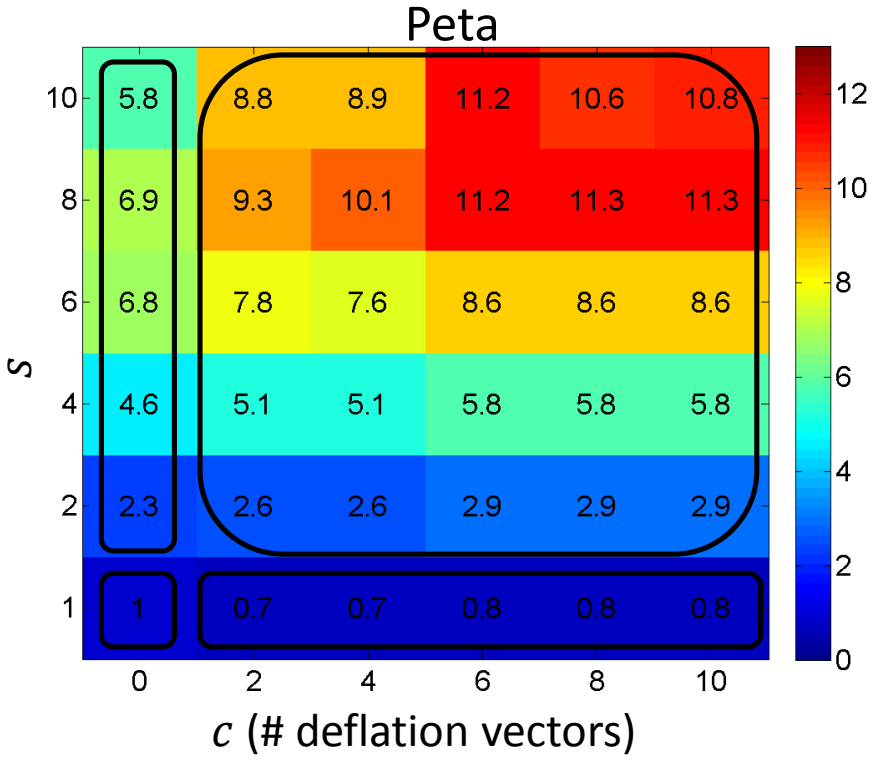


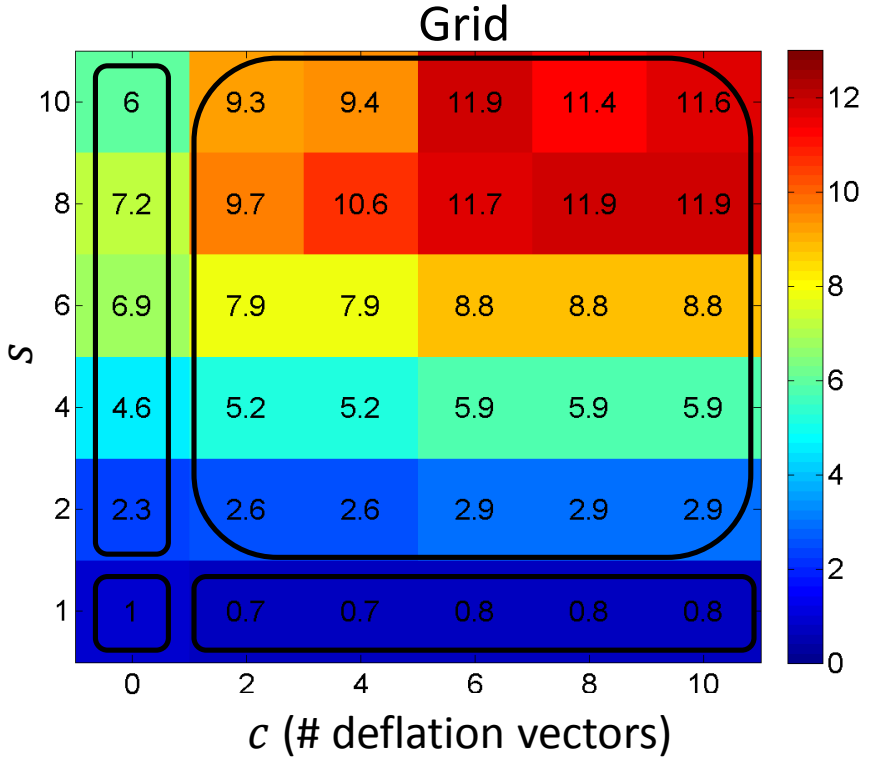
Peta: Overlapping communication and computation decreases cost of increasing c, so deflation results in performance improvement

Grid: Overlapping communication and computation doesn't change anything, since extremely communication bound

Total Speedup with Overlap (Monomial)

Total speedup = (speedup per iteration) × (number of iterations(Monomial))



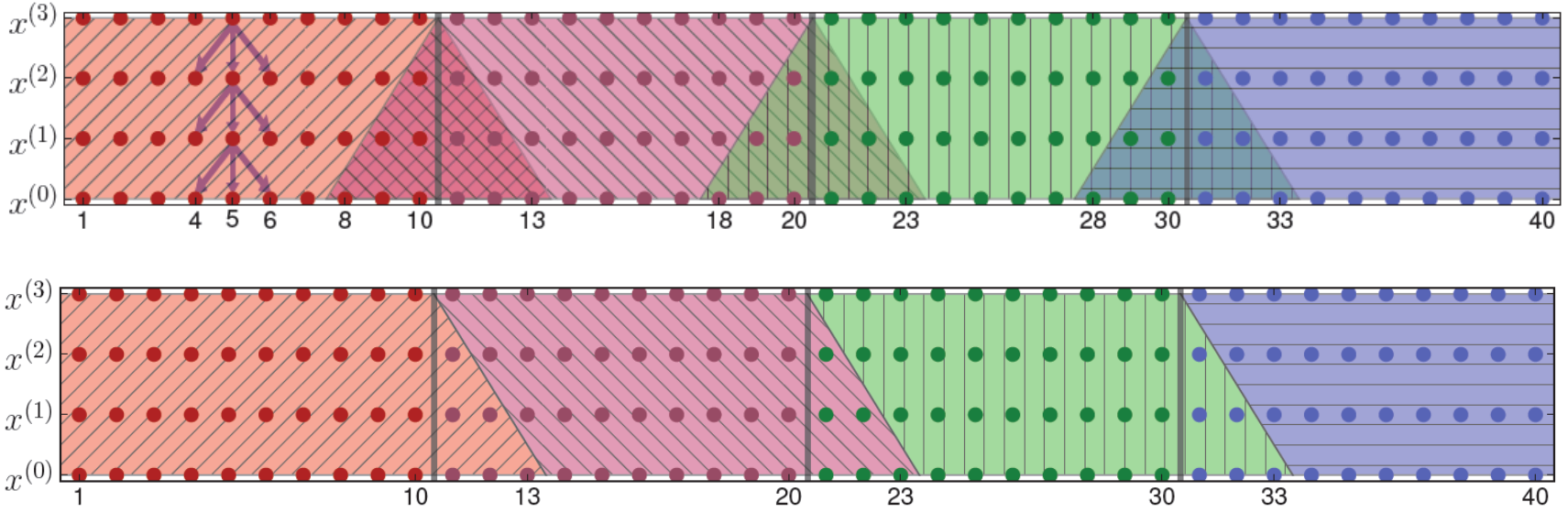


Peta: Overlapping communication and computation decreases cost of increasing c; deflation results in speedup (but since convergence decreases s, best speedup at s=8.

Grid: Overlapping communication and computation doesn't change anything, since extremely communication bound

The Matrix Powers Kernel

- Compute dependencies up front for computing $Av, A^2v, \ldots, A^sv$
  - s steps of the transitive closure of A
  - Only need to read A once assuming A is well-partitioned



Figures: [MHDY09]