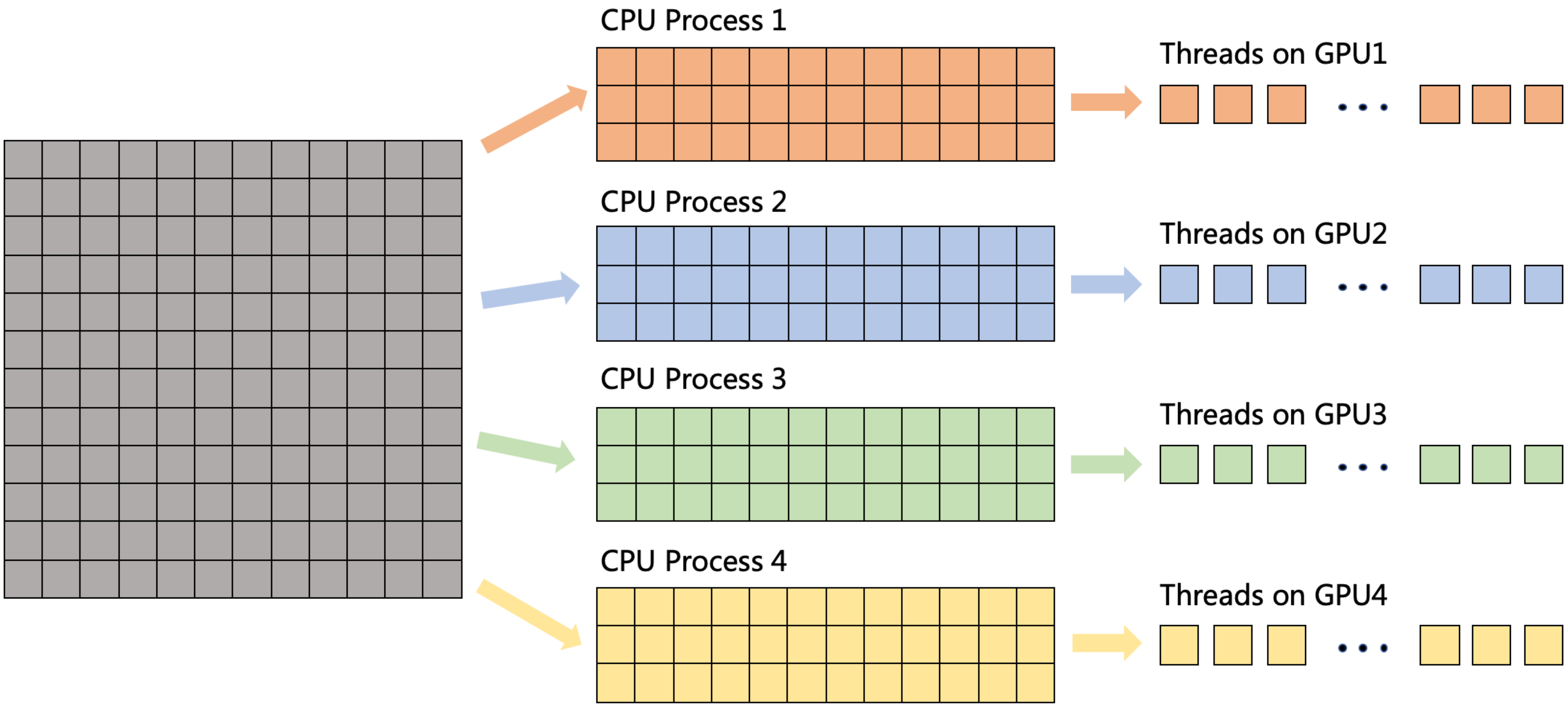
Section 3. Parallel Krylov Subspace Methods

Parallel Matrix Data Layout



Basic Ideas on Reducing Communication

So communication could be more costly compared to computation. How can we get around?

Num of messages

Reduce total number of messages needed by better organizing algorithms, combining messages, ……

Size of messages

Reduce amount of data that need to be moved by better iterative algorithms, better partitioning, …….

Comm. hiding

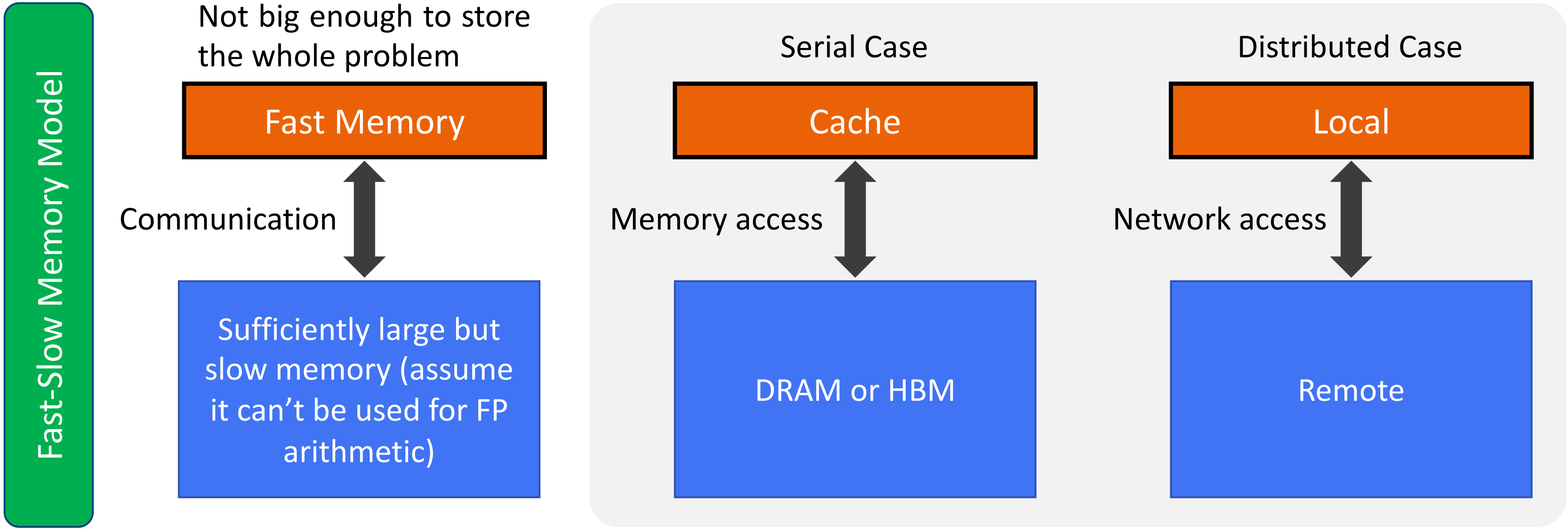
Hide communication behind computation by aligning them in a smart way, ……

Better network

Use better inter-connecting network with high throughput, ……

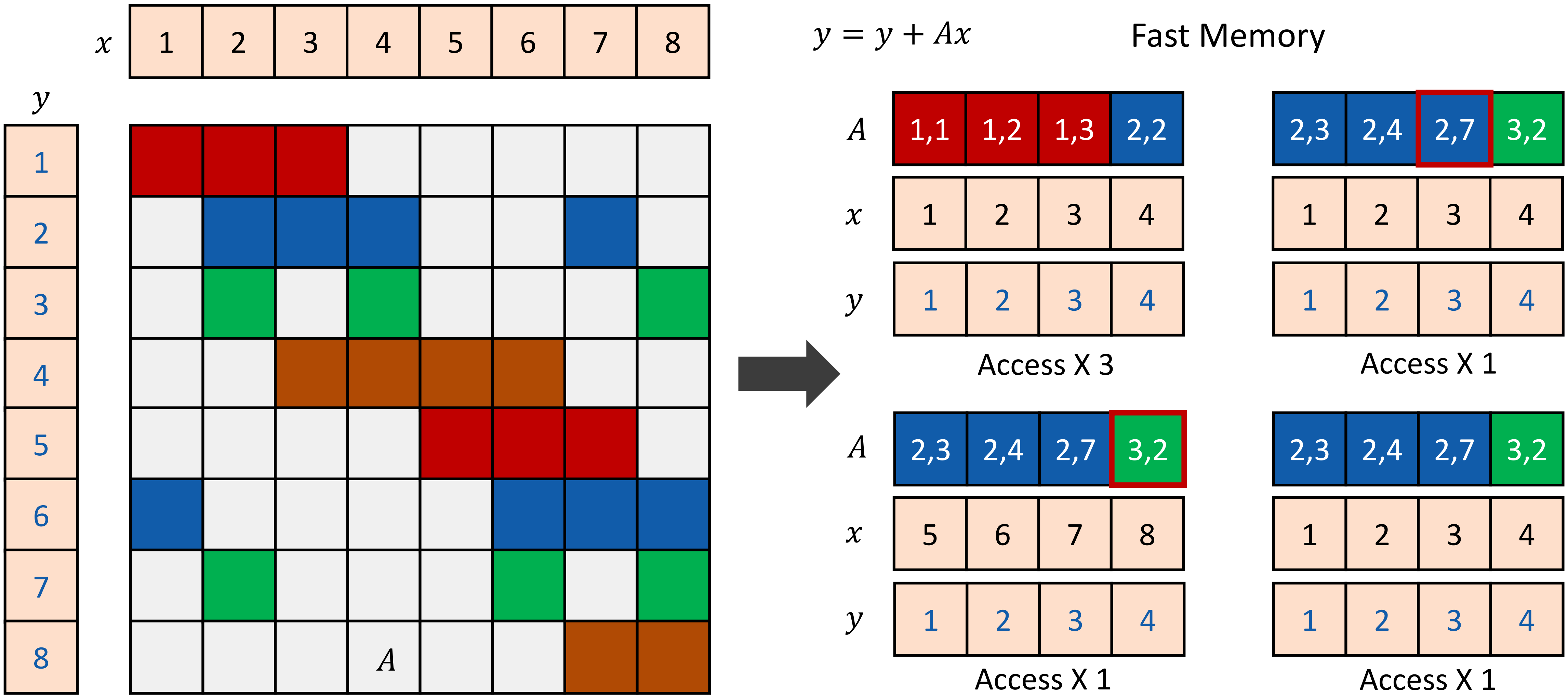
Communication Performance Model

Communication Time = **Latency** + Num of Bytes Moved ÷ **Bandwidth**



Ref: CS267 lecture notes on J. Demmel's webpage: https://people.eecs.berkeley.edu/~demmel/

SpMV, Overly Simplified Case



GMRES Method, Revisited

- The generalized minimum residual (GMRES) method finds:

$$\min_{e \in \mathcal{K}_m(A,r)} \|r - Ae\|_0$$

in the Krylov subspace

$$\mathcal{K}_m(A,r) := \mathrm{span}\{r, Ar, A^2 r, \ldots, A^{m-1} r\}$$

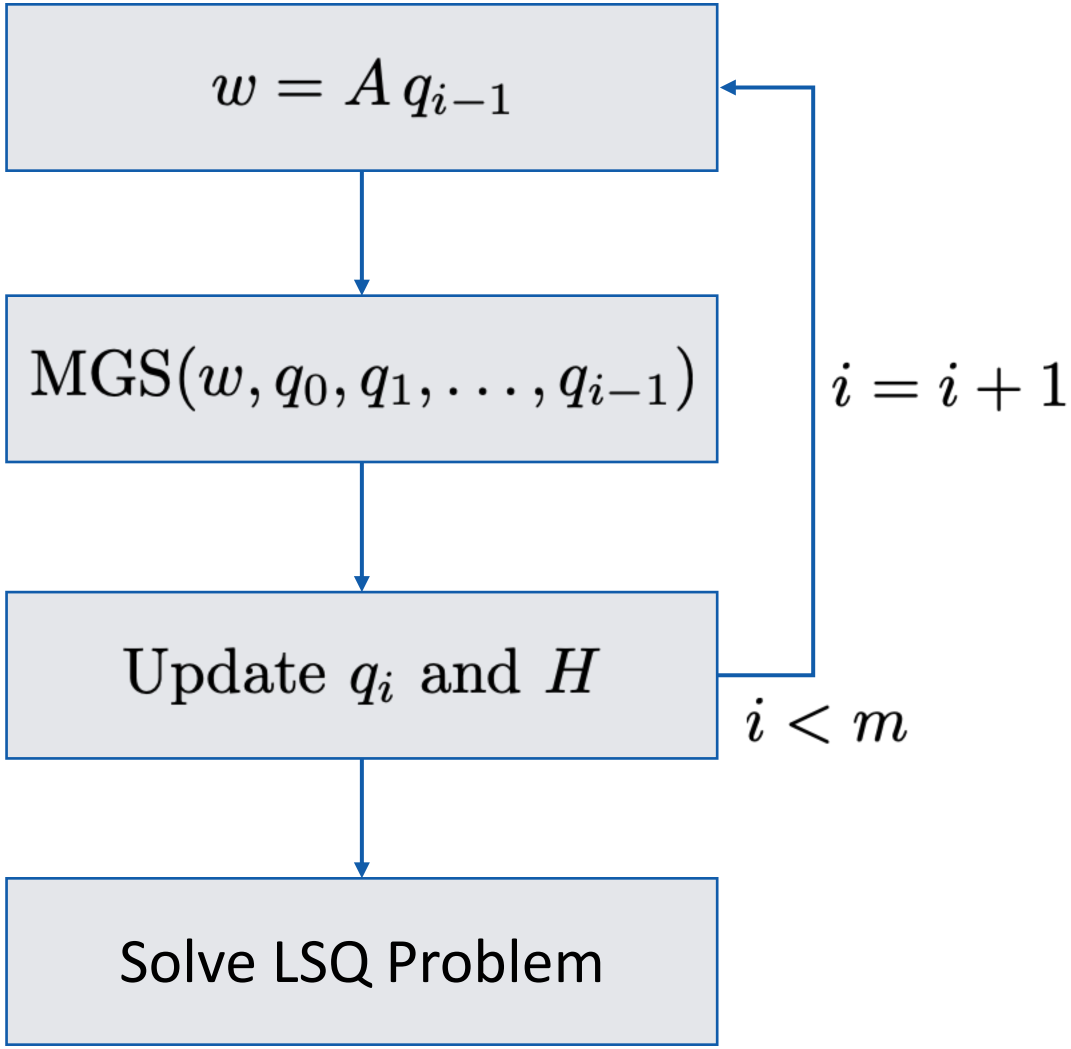
- We form an orthonormal basis of the Krylov subspace

$$\mathcal{K}_m := \mathrm{span}\{q_1, q_2, \ldots, q_m\}$$

- By applying the modified Gram-Schmidt (MGS) algorithm, we form $\overline{H}_m$ and then solve the least squares (LSQ) problem with $\overline{H}_m$

Q: Remember why we use such implementation? We tried to: (1) ease numerical instability; (2) use an iterative procedure that can stop at any time. However, communication was never considered!

Data Movement in GMRES



- Analyzing data movement is difficult
  - Parallel architectures
  - Parallel data layout
  - Parallel algorithm

- SpMV — No chance for data reuse
  - Words moved $\sim O(m \cdot nnz)$
  - Number of messages $\sim O(m)$

- MGS — Iterative procedure
  - Words moved $\sim O(m^2 \cdot n)$
  - Number of messages $\sim O(m^2 \cdot \log P)$

Communication-Avoiding GMRES

$$W = [Aq_0, A^2 q_0, \ldots, A^m q_0]$$

$\downarrow$

$$[Q, R] = \mathrm{TSQR}(W)$$

$\downarrow$

$$\text{Build } H$$

$\downarrow$

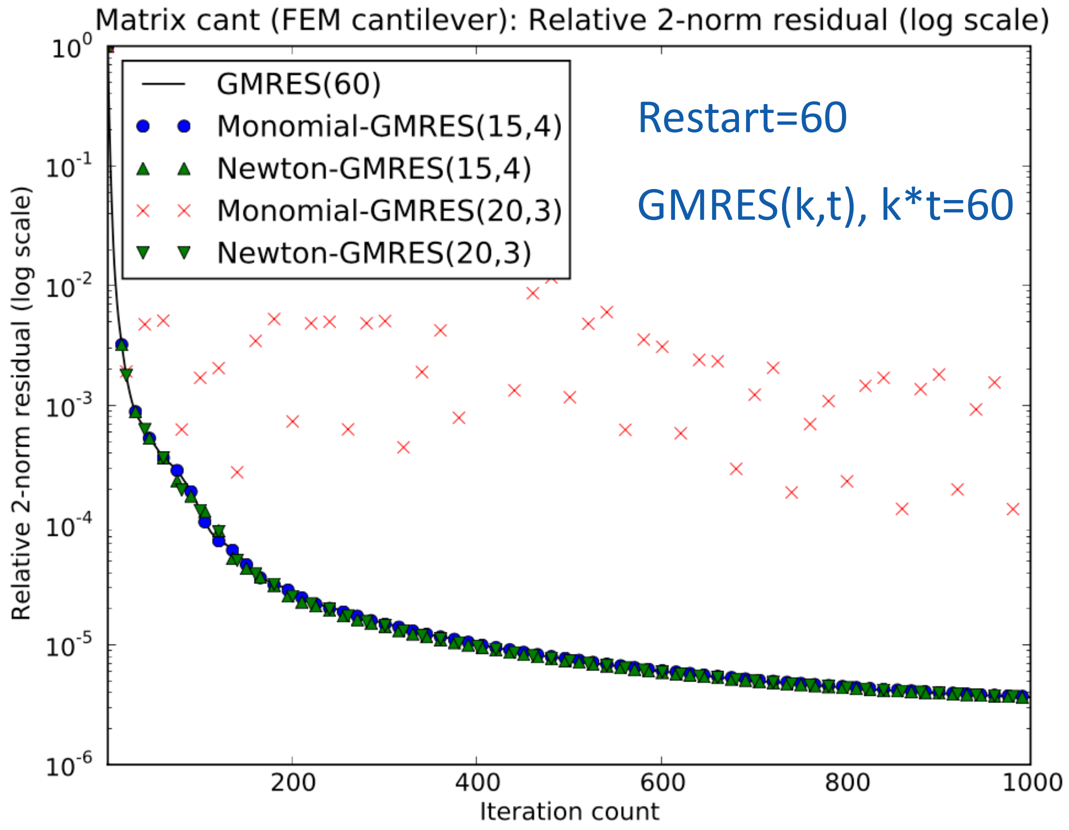
Solve LSQ Problem

- Reorganize the algorithm
  - Identical mathematical method (with exact FP)
  - Use the matrix powers kernel
  - Use QR factorization instead of MGS

- Matrix Powers Kernel
  - Words moved $\sim O(nnz)$
  - Number of messages $\sim O(1)$

- TSQR
  - Words moved $\sim O(m \cdot n)$
  - Number of messages $\sim O(\log P)$

Performance of CA-GMRES



- The "easy" implementation of CA-GMRES is not stable because the matrix powers kernel may produce linearly dependent vectors
- Use the Newton basis (shifted polynomials based on the eigenvalues of the upper Hessenberg matrix) proposed by Bai, Hu, and Reichel, 1994

$$W = [(A - \lambda_1 I)q_0, \ldots, \Pi_{j=1}^{m}(A - \lambda_j I)q_0]$$

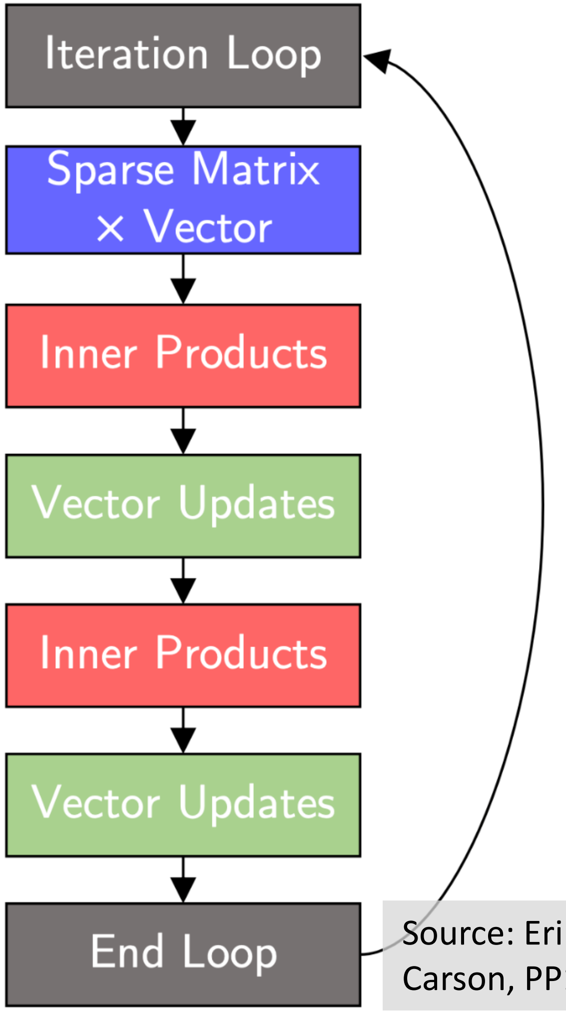
Source: Marghoob Mohiyuddin, Mark Hoemmen, James Demmel, and Katherine Yelick. Minimizing communication in sparse matrix solvers. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC 2009).

Conjugate Gradient Method, Revisited

SpMV is the most expensive part: more floating-point calculations required and not cache-friendly (memory-bound)

Algorithm 2: Conjugate gradient method

```
%% Given an initial guess u and a tolerance ε;
r ← f − Au,  p ← r;
while ‖r‖ > ε
      α ← (r, r)/(Ap, p);
      ũ ← u + α p;
      r̃ ← r − αAp;
      β ← (r̃, r̃)/(r, r);
      p̃ ← r̃ + β p;
      Update:  u ← ũ,  r ← r̃,  p ← p̃;
end
```



Source: Erin Carson, PP18

Inner products are expensive for communication: not much computation cost but global all-reduce is necessary (communication-bound)

Three-Term Recurrence CG

M. Hoemmen 2010

- Based on the three-term recurrence formulation for residuals

$$r_{k+1} = \rho_k(r_k - \gamma_k A r_k) + (1 - \rho_k) r_{k-1}$$

and the residuals are orthogonal to each other, we have

$$\gamma_k = \frac{(r_k, r_k)}{(A r_k, r_k)}$$

$$\rho_k = \left(1 - \frac{\gamma_k}{\gamma_{k-1}} \frac{(r_k, r_k)}{(r_{k-1}, r_{k-1})} \frac{1}{\rho_{k-1}}\right)^{-1}$$

- We can derive a new recurrence relation

$$x_{k+1} = \rho_k(x_k + \gamma_k r_k) + (1 - \rho_k) x_{k-1}$$

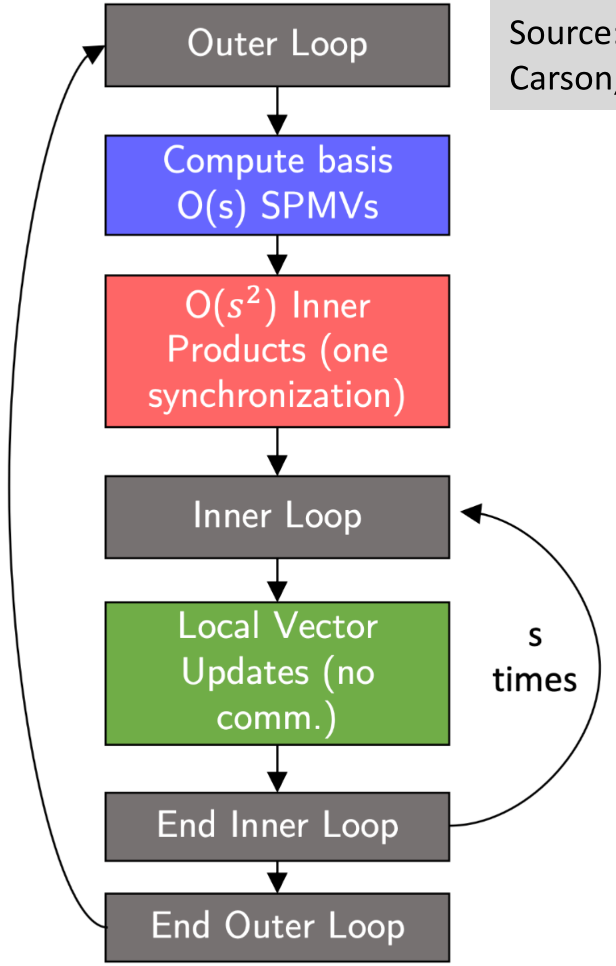
This formulation (usually known as CG3) does not involve the conjugate directions. This is desirable for deriving s-step CG algorithms.

Ref: Y. Saad, "Iterative Methods for Sparse Linear Systems", SIAM, Philadelphia, Second Ed., 2003

s-Step CG3 Method

Algorithm 19: s-Step conjugate gradient method

%% Given an initial guess u_1 and a tolerance ε;
$r_1 \leftarrow f - \mathcal{A} u_1$;
for k = 0:MaxIter
    for j = 1:s
        $w_{sk+j} \leftarrow \mathcal{A} r_{sk+j}$; %% P2P communication
        $\mu_{sk+j} \leftarrow (r_{sk+j}, r_{sk+j})$; %% Test for convergence; All_reduce
        $\nu_{sk+j} \leftarrow (w_{sk+j}, r_{sk+j})$;
        $\gamma_{sk+j} \leftarrow \mu_{sk+j} / \nu_{sk+j}$;
        if $sk + j == 1$
            $\rho_{sk+j} \leftarrow 1$;
        else
            $\xi_1 \leftarrow \gamma_{sk+j} / \gamma_{sk+j-1}$;
            $\xi_2 \leftarrow \mu_{sk+j} / \mu_{sk+j-1}$;
            $\rho_{sk+j} \leftarrow (1 - \xi_1 \xi_2 / \rho_{sk+j-1})^{-1}$;
        end
        $x_{sk+j+1} \leftarrow \rho_{sk+j}(x_{sk+j} + \gamma_{sk+j} r_{sk+j}) + (1 - \rho_{sk+j}) x_{sk+j-1}$;
        $r_{sk+j+1} \leftarrow \rho_{sk+j}(r_{sk+j} - \gamma_{sk+j} w_{sk+j}) + (1 - \rho_{sk+j}) r_{sk+j-1}$;
    end
end



Source: Erin Carson, PP18

Ref: Mark F. Hoemmen, Communication-avoiding Krylov subspace methods, Ph.D. thesis, 2010

Communication-Avoiding CG

M. Hoemmen 2010

- We have the recurrence relation for residual

$$r_{sk+j+1} = \rho_{sk+j}\left(r_{sk+j} - \gamma_{sk+j} A r_{sk+j}\right) + (1-\rho_{sk+j}) r_{sk+j-1}$$

- Rearrange the terms as follows:

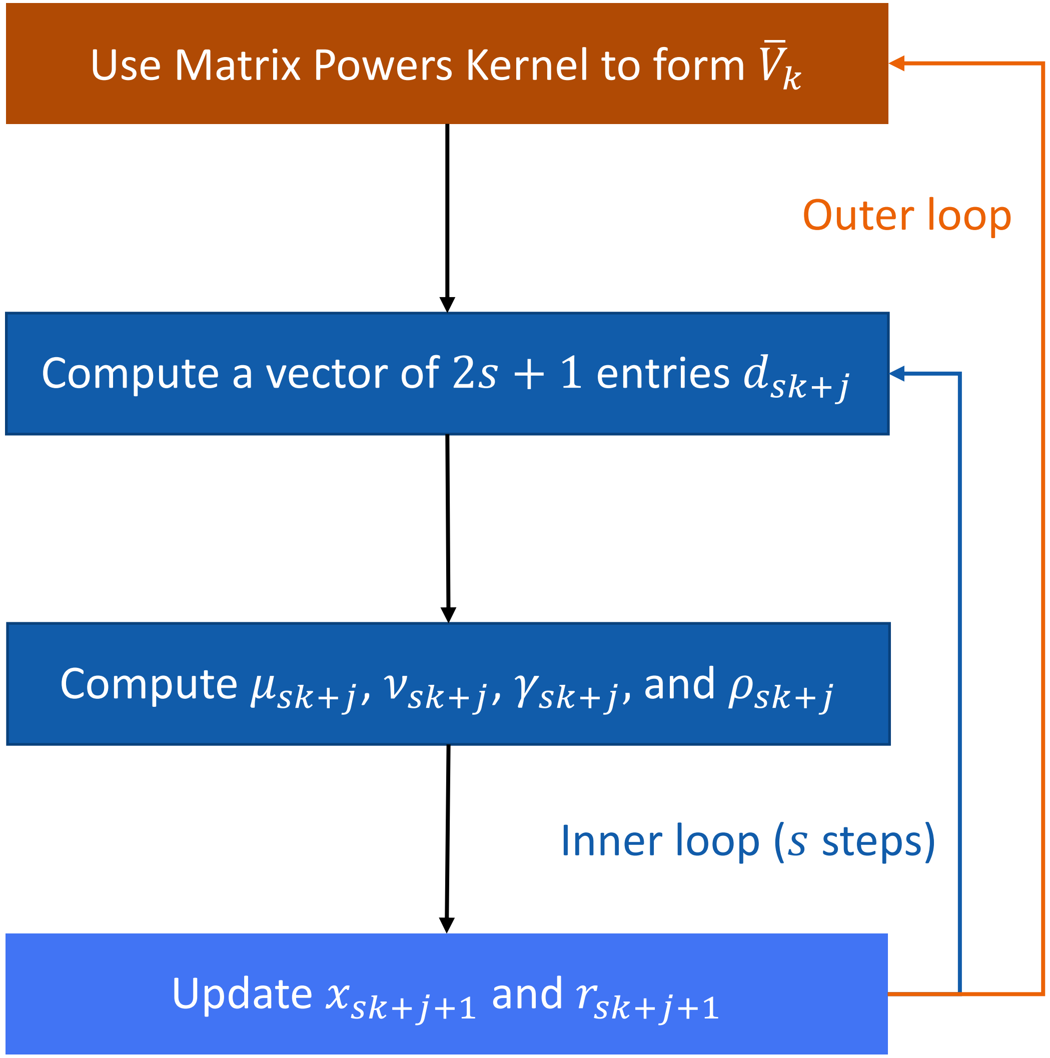
$$A r_{sk+j} = \frac{1-\rho_{sk+j}}{\rho_{sk+j}\gamma_{sk+j}} r_{sk+j-1} + \frac{1}{\rho_{sk+j}\gamma_{sk+j}} r_{sk+j} - \frac{1}{\rho_{sk+j}\gamma_{sk+j}} r_{sk+j+1}$$

- Write the recurrence in terms of matrix form:

$$\boxed{A\underbrace{\left[r_{sk+1},\ldots,r_{sk+s}\right]}_{R_k} = \frac{1-\rho_{sk}}{\rho_{sk}\gamma_{sk}} r_{sk} e_1^T + \underbrace{\left[r_{sk+1},\ldots,r_{sk+s+1}\right]}_{\bar{R}_k}\bar{T}_k}$$

where $\bar{T}_k$ is a $(s+1)\times s$ tridiagonal matrix in terms of $\rho_{sk+1},\ldots,\rho_{sk+s},\gamma_{sk+1},\ldots,\gamma_{sk+s}$

From s-Step CG To CA-CG



$$\bar{V}_k := \left[v_{sk+1}, \ldots, v_{sk+s}\right]$$

$$\{v_{sk+i}\}_{i=1:s+1} = \operatorname{span}\{r_{sk+1}, Ar_{sk+1}, \ldots, A^s r_{sk+1}\}$$

is a basis of the Krylov subspace

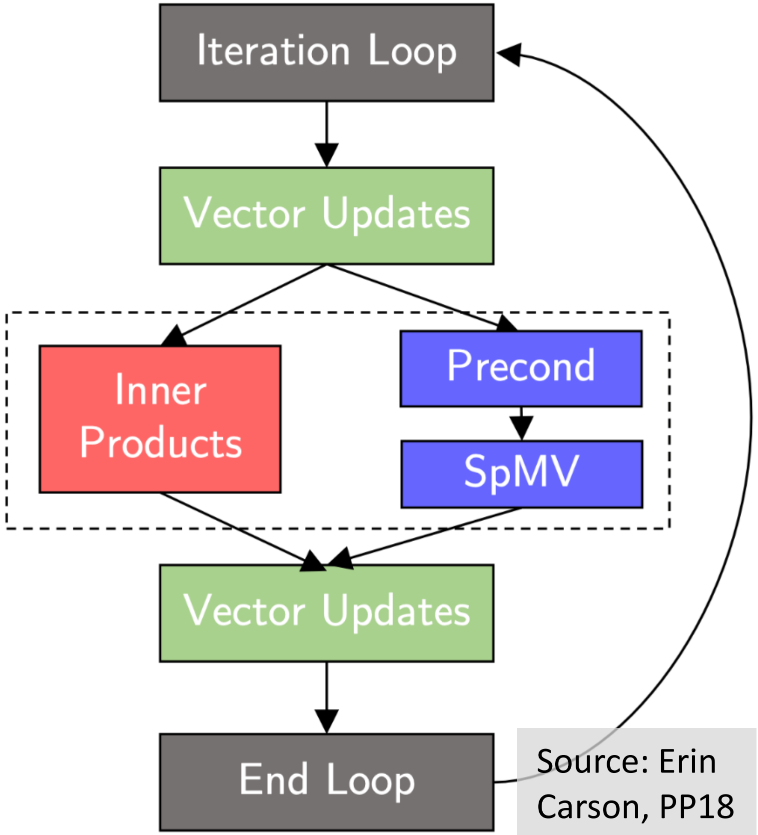
$$\boxed{w_{sk+j} := A\,r_{sk+1} = \left[R_{k-1}, \bar{V}_k\right] d_{sk+j}}$$

- The matrix powers kernel only needs to load the coefficient matrix once
- In exact arithmetic, the algorithm produces the same results as the standard CG
- Further improvement by using an inner product coalescing kernel Sec 5.4.4, M. Hoemmen 2010

Pipelined Conjugate Gradient Method

Algorithm 20: Pipelined conjugate gradient method

%% Given an initial guess u and a tolerance ε;
$r \leftarrow f - \mathcal{A}u,\ p \leftarrow r$;
$s \leftarrow \mathcal{A}p,\ w \leftarrow \mathcal{A}r,\ z \leftarrow \mathcal{A}w$;
$\alpha \leftarrow (r,r)/(p,s)$;
while $\|r\| > \varepsilon$
    $\tilde{u} \leftarrow u + \alpha p$;
    $\tilde{r} \leftarrow r - \alpha s$;
    $\tilde{w} \leftarrow w - \alpha z$;
    $\tilde{z} \leftarrow \mathcal{A}\tilde{w}$; %% SpMV, asymc
    $\beta \leftarrow (\tilde{r},\tilde{r})/(r,r)$;
    $\alpha \leftarrow \frac{(\tilde{r},\tilde{r})}{(w,r)-(\beta/\alpha)(\tilde{r},\tilde{r})}$;
    $\tilde{p} \leftarrow \tilde{r} + \beta p$;
    $\tilde{s} \leftarrow \tilde{w} + \beta s$;
    $\tilde{z} \leftarrow \tilde{z} + \beta z$; %% Results of SpMV needed here!
    Update: $u \leftarrow \tilde{u},\ r \leftarrow \tilde{r},\ p \leftarrow \tilde{p},\ w \leftarrow \tilde{w},\ s \leftarrow \tilde{s},\ z \leftarrow \tilde{z}$;
end



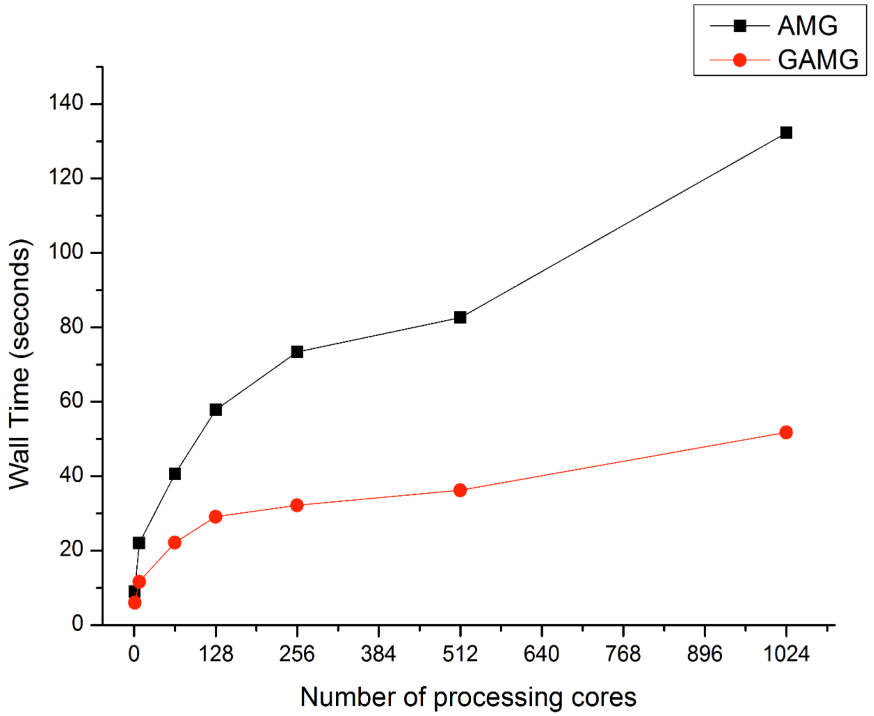
Source: Erin Carson, PP18

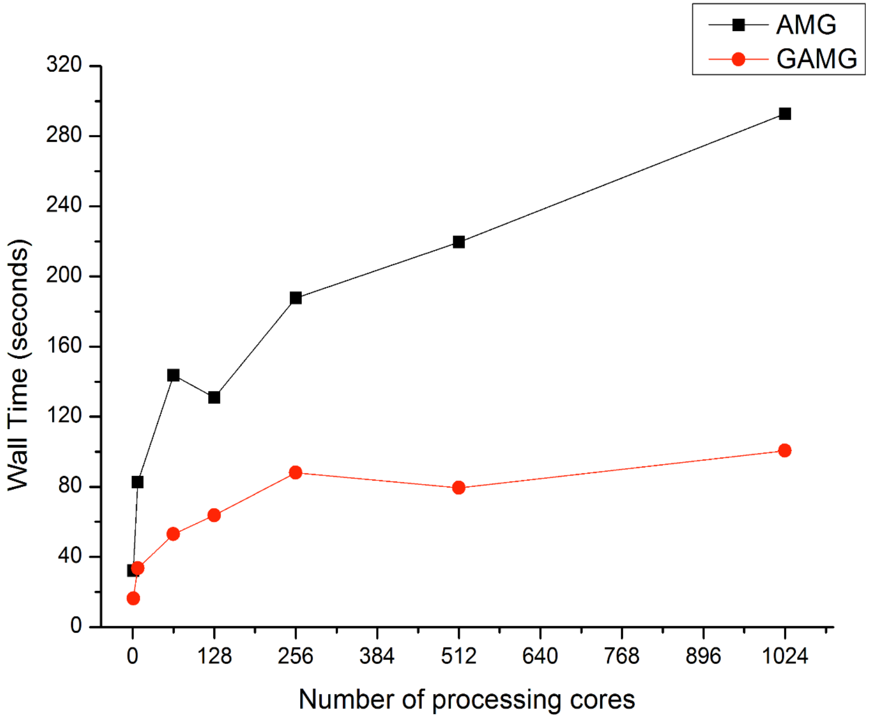
Ref: Ghysels, Pieter and Wim Vanroose. "Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm." Parallel Comput. 40 (2014): 224-238.

Section 4. KSM and Preconditioning Methods

Taking Preconditioning Into Account

- Note: Preconditioning might take more time than other parts in practice
- Parallelization must take preconditioning part into account





Source: A stable and scalable hybrid solver for rate-type non-Newtonian fluid models, Y.-J. Lee, W. Leng, and C.-S. Zhang, Journal of Computational and Applied Mathematics, 300, 103–118 (07/2016).

Convergence Result of KSM

Theorem 2.43 (Convergence of KSM in Hilbert spaces). Let $\mathcal{A} : \mathscr{V} \mapsto \mathscr{V}$ be a symmetric isomorphism. The minimum residual method satisfies the following estimate:

$$\left\|\mathcal{A}(u - u^{(m)})\right\| \leqslant 2\delta^m \left\|\mathcal{A}(u - u^{(0)})\right\|, \tag{2.44}$$

where $0 < \delta < 1$ only depends on $\kappa(\mathcal{A})$. Moreover, if $\mathcal{A}$ is positive-definite, then the conjugate gradient method satisfies that

$$\left\|u - u^{(m)}\right\|_{\mathcal{A}} \leqslant 2\delta^m \left\|u - u^{(0)}\right\|_{\mathcal{A}}, \tag{2.45}$$

where $\delta = (\sqrt{\kappa(\mathcal{A})} - 1)/(\sqrt{\kappa(\mathcal{A})} + 1)$.

- Q: Can we apply KSM to infinite dimensional problems?
- The above convergence estimates do not depend on dimensionality

More General Setting for KSM

- We consider a more general and more natural setting:

$$\mathcal{A} : \mathcal{V} \mapsto \mathcal{W}, \quad \text{where } \mathcal{V} \text{ and } \mathcal{W} \text{ are both separable Hilbert spaces}$$

- Typically, we have $\mathcal{V} \subset \mathcal{W}, \quad \text{e.g. } \mathcal{W} = \mathcal{V}'$

- Note: Apparently, KSM cannot be directly applied in this setting anymore!

- Need to construct an isomorphism $\mathcal{B} : \mathcal{V}' \mapsto \mathcal{V}$

- Define a Riesz operator

$$\text{For any given } f \in \mathcal{V}' : \quad (\mathcal{B}f, v)_{\mathcal{V}} = \langle f, v \rangle, \quad \forall v \in \mathcal{V}$$

- Preconditioned system $\boxed{\mathcal{B}\mathcal{A}u = \mathcal{B}f}$

Source: Mardal and Winther, NLAA 2011

Condition Number Analysis

- Convergence results similar to Theorem 2.43 can be obtained:

$$\langle \mathcal{A}(u-u^{(m)}), \mathcal{B}\mathcal{A}(u-u^{(m)})\rangle^{1/2} \le 2\delta^m \langle \mathcal{A}(u-u^{(0)}), \mathcal{B}\mathcal{A}(u-u^{(0)})\rangle^{1/2}$$

  where δ depends on $\kappa(\mathcal{B}\mathcal{A})$ only.

- From symm, continuity, and inf-sup condition of $a[\cdot,\cdot]$, we get boundedness of condition number:

$$(\mathcal{B}\mathcal{A}u, v)_{\mathcal{V}} = \langle \mathcal{A}u, v\rangle = a[u,v] = (u, \mathcal{B}\mathcal{A}v)_{\mathcal{V}}, \quad u, v \in \mathcal{V}$$

$$\|\mathcal{B}\mathcal{A}\|_{\mathcal{L}(\mathcal{V};\mathcal{V})} = \sup_{v\in\mathcal{V}} \frac{|(\mathcal{B}\mathcal{A}v, v)_{\mathcal{V}}|}{\|v\|_{\mathcal{V}}^2} = \sup_{v\in\mathcal{V}} \frac{a[v,v]}{\|v\|_{\mathcal{V}}^2} \le C_a$$

$$\|(\mathcal{B}\mathcal{A})^{-1}\|_{\mathcal{L}(\mathcal{V};\mathcal{V})}^{-1} = \inf_{v\in\mathcal{V}} \frac{\|\mathcal{B}\mathcal{A}v\|_{\mathcal{V}}}{\|v\|_{\mathcal{V}}} = \inf_{v\in\mathcal{V}} \sup_{u\in\mathcal{V}} \frac{(\mathcal{B}\mathcal{A}v, u)_{\mathcal{V}}}{\|v\|_{\mathcal{V}}\|u\|_{\mathcal{V}}} = \inf_{v\in\mathcal{V}} \sup_{u\in\mathcal{V}} \frac{a[v,u]}{\|v\|_{\mathcal{V}}\|u\|_{\mathcal{V}}} \ge \alpha$$

Second-order Elliptic Problem

- Assume the diffusion coefficient is uniformly bounded

$$\mu(x) \in \mathbb{R}^{d\times d} \quad \Longrightarrow \quad c|\xi|^2 \le \xi^T \mu(x)\xi \le C|\xi|^2, \quad x \in \Omega,\ \xi \in \mathbb{R}^d.$$

- Consider the linear operator

$$\mathcal{A} : H_0^1(\Omega) \mapsto H^{-1}(\Omega) \ : \quad \langle \mathcal{A}u, v\rangle = a[u,v] := \int_\Omega (\mu(x)\nabla u)\cdot \nabla v\, dx.$$

- Define a natural preconditioner

$$\big(\mathcal{B}f, v\big)_{H_0^1(\Omega)} := \big(\nabla(\mathcal{B}f), \nabla v\big)_{0,\Omega} = \langle f, v\rangle \quad \Longrightarrow \quad \mathcal{B} := (-\Delta)^{-1}$$

- Uniform convergence

$$\kappa(\mathcal{B}\mathcal{A}) = \|\mathcal{B}\mathcal{A}\|_{\mathcal{L}(H_0^1(\Omega);H_0^1(\Omega))} \|(\mathcal{B}\mathcal{A})^{-1}\|_{\mathcal{L}(H_0^1(\Omega);H_0^1(\Omega))} \le \frac{C}{c}$$

Constructing Natural Preconditioners

- Define an appropriate inner product $(\cdot,\cdot)_{\mathcal{V}}$
- Establish the inf-sup condition:

$$\sup_{v\in\mathcal{V}} \frac{a[u,v]}{\|v\|_{\mathcal{V}}} \geq \alpha\|u\|_{\mathcal{V}}, \quad \forall u \in \mathcal{V}$$

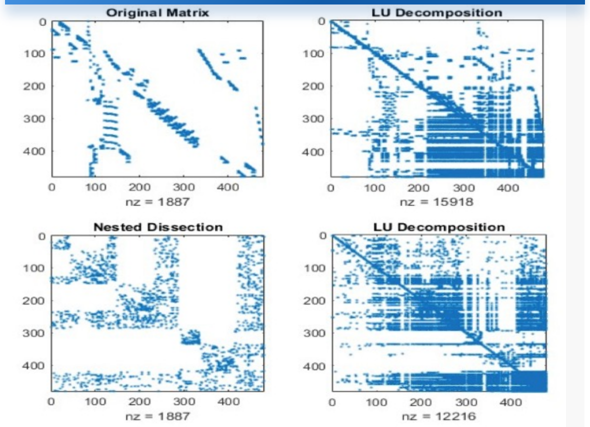
- Define the Reisz operator

$$(\mathcal{B}f, v)_{\mathcal{V}} = \langle f, v\rangle, \quad \forall v \in \mathcal{V}$$

- The preconditioned system $\mathcal{BA}$ is symmetric with respect to $(\cdot,\cdot)_{\mathcal{V}}$ and well-conditioned
- Construct a discretization which satisfies the corresponding discrete inf-sup condition
- Define a spectrally equivalent discrete preconditioner

Preconditioning Techniques

基于代数的方法

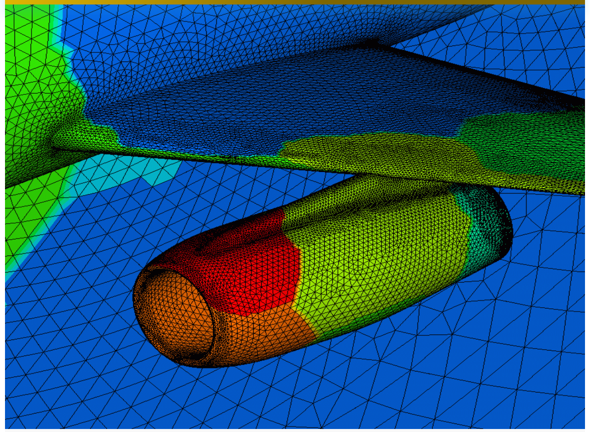


01.

LU, ILU, SAI, …

纯代数，通用性强，稳健性高，用户友好；效率一般不高，并行可扩展性较差。

基于区域分解的方法

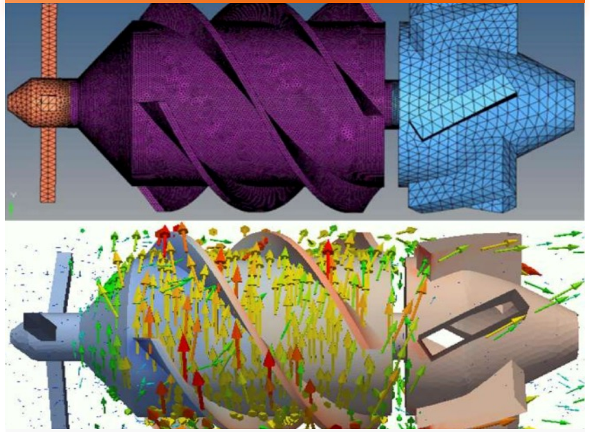


02.

DDM, RAS, FETI-DP, …

可以基于网格进行，效率高，通用性较强，可扩展性较强；难以兼顾通用性与最优性。

基于物理的方法



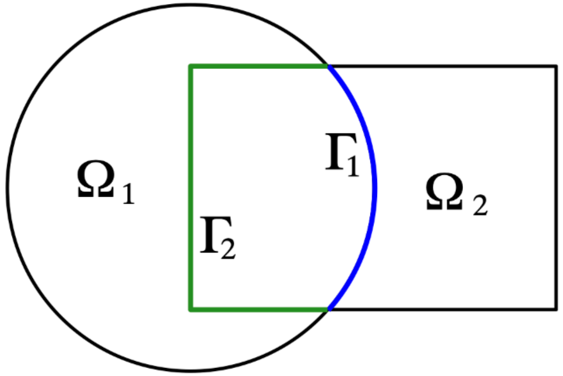
03.

Block Preconditioners

算法灵活，基于成熟算法开发，效率高，可扩展性强；通用性弱，用户友好度差。

简单、易用 ⟷ **通用性强、可扩展** ⟷ **有针对性、效率高**

Domain Decomposition Method



Divide and conquer

$$\begin{cases} \mathcal{A}u_1^{(m+1)} = f & \text{in } \Omega_1, \\ u_1^{(m+1)} = u^{(m)} & \text{on } \Gamma_1, \\ u_1^{(m+1)} = 0 & \text{on } \partial\Omega_1 \backslash \Gamma_1, \end{cases}$$

$$\begin{cases} \mathcal{A}u_2^{(m+1)} = f & \text{in } \Omega_2, \\ u_2^{(m+1)} = g^{(m)} & \text{on } \Gamma_2, \\ u_2^{(m+1)} = 0 & \text{on } \partial\Omega_2 \backslash \Gamma_2. \end{cases}$$

$\Rightarrow$

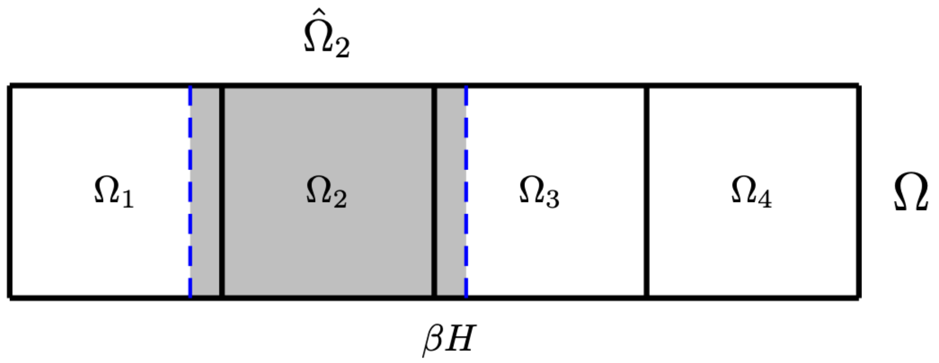
How to set boundary conditions for subdomains?

$$\begin{cases} g^{(m)} = u^{(m)} & \text{Additive method} \\ g^{(m)} = u_1^{(m+1)} & \text{Multiplicative method} \end{cases}$$

$$u^{(m+1)}(x) := \begin{cases} u_2^{(m+1)}, & \text{if } x \in \Omega_2; \\ u_1^{(m+1)}, & \text{if } x \in \Omega \backslash \Omega_2. \end{cases}$$

DDM is widely used in, e.g., parallel computing, solving multi-physics problems, as well as preconditioners!

Overlapping DDM for Linear Systems



$G := \{1, 2, \ldots, N\}$ grid points

$G = \hat{G}_1 \bigcup \hat{G}_2 \bigcup \cdots \bigcup \hat{G}_n$ subdomain grid points

$I_i \in \mathbb{R}^{N \times N_i}$ injection, natural embedding:

$$(I_i \vec{v}_i)_k = \begin{cases} (\vec{v}_i)_k, & \text{if } k \in \hat{G}_i; \\ 0, & \text{if } k \in G \backslash \hat{G}_i. \end{cases}$$

- Form subdomain problems and choose subdomain solvers

$$A_i := I_i^T A I_i, \quad B_i := I_i A_i^{-1} I_i^T$$

- Apply the DDM idea as a linear solver (preconditioner)

Additive Schwarz method

$$B_{\mathrm{as}} := \sum_{i=1}^{n} B_i = \sum_{i=1}^{n} I_i A_i^{-1} I_i^T$$

Multiplicative Schwarz method

$$I - B_{\mathrm{ms}} A := \prod_{i=n}^{1} (I - B_i A).$$

Convergence of Overlapping DDM

Theorem 2.49 (Effect of DD preconditioner)**.** The condition number of AS domain decomposition method is independent of the mesh size h and satisfies

$$\kappa(B_{\mathrm{as}}A) \leqslant CH^{-2}(1+\beta^{-2}),$$

where H is size of domain partitions, βH characterizes size of the overlaps, and C is a constant independent of mesh sizes.

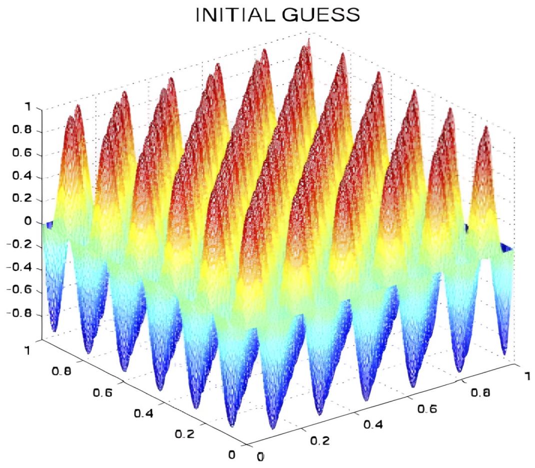
- Introduce a coarse space $V_0 \subset V$ and a corresponding coarse-level solver, i.e.

$$B_{\mathrm{as},2} := I_0 A_0^{-1} I_0^T + \sum_{i=1}^{n} I_i A_i^{-1} I_i^T$$

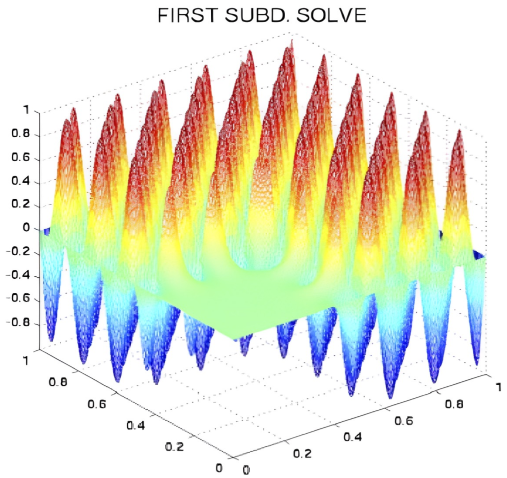
- The two-level additive Schwarz preconditioner is uniform with respect to subdomain size

$$\kappa(B_{\mathrm{as},2}A) \lesssim 1+\beta^{-1}$$

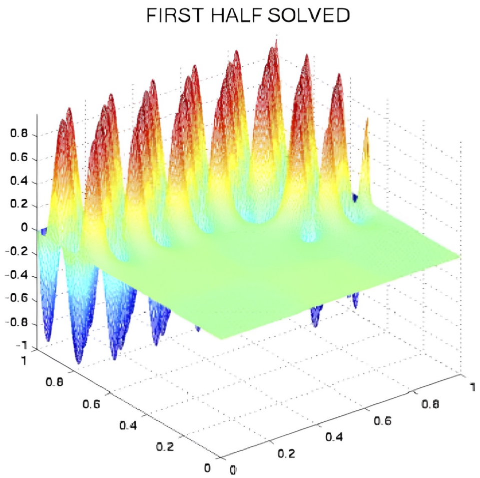
Smoothing and CGC in TG DDM



Solve the subproblem on the first subdomain

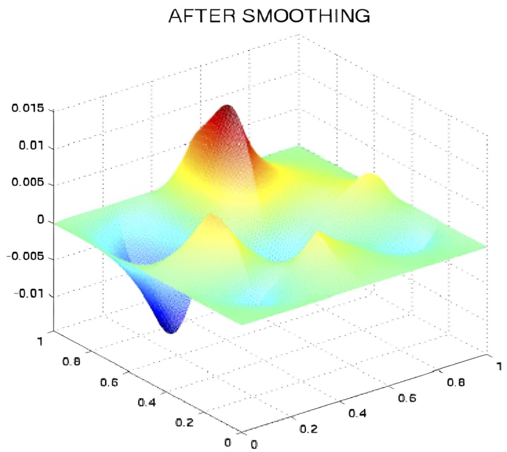


Solve the subproblem on the second subdomain

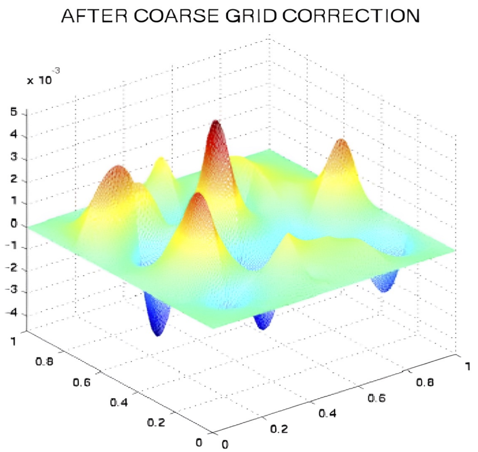


Solve the Poisson's equation using the two-level domain decomposition method:
(Upper) smoothing;
(Lower) CGC

After one iteration of DDM on four subdomains



After coarse grid correction



When Is Coarse Approximation Good

- Suppose we have fine and coarse finite element solutions: $u_h \in V_h, \quad u_H \in V_H, \quad V_H \subset V_h$
- Galerkin orthogonality: $a[u_h - u_H, v_H] = 0, \quad \forall\, v_H \in V_H$
- Duality argument

$$\begin{cases} -\Delta w = u_h - u_H & \text{in } \Omega, \\ w = 0 & \text{on } \partial\Omega. \end{cases} \xrightarrow{\text{assuming full elliptic regularity}} \|w\|_2 \leqslant C\|u_h - u_H\|_0$$

$$\Longrightarrow \|u_h - u_H\|_0^2 = a[w, u_h - u_H] = a[w - w_H, u_h - u_H]$$
$$\leqslant \|w - w_H\|\,\|u_h - u_H\| \lesssim H|w|_2\,\|u_h - u_H\|\,.$$

- Difference between fine and coarse approximations

$$\boxed{\|u_h - u_H\|_0 \lesssim H\,\|u_h - u_H\| \lesssim H\,\|u_h\|\,.}$$

Coarse solution is a good approximation if fine solution is smooth!

Twogrid Method

- The multigrid V-cycle:

Algorithm (One iteration of multigrid method $\vec{u}_l = MG(l, \vec{f}_l, \vec{u}_l)$)

1. Pre-smoothing: $\vec{u}_l \leftarrow \vec{u}_l + \frac{1}{2} D_l^{-1} \big(\vec{f}_l - A_l \vec{u}_l\big)$.
2. Restriction: $\vec{r}_{l-1} \leftarrow R_{l,l-1} \big(\vec{f}_l - A_l \vec{u}_l\big)$.
3. Coarse-grid correction: If $l = 1$, $\vec{e}_{l-1} \leftarrow A_{l-1}^{-1} \vec{r}_{l-1}$; $\vec{e}_{l-1} \leftarrow MG(l-1, \vec{r}_{l-1}, \vec{0}_{l-1})$, otherwise.
4. Prolongation: $\vec{u}_l \leftarrow \vec{u}_l + P_{l-1,l} \vec{e}_{l-1}$.
5. Post-smoothing: $\vec{u}_l \leftarrow \vec{u}_l + \frac{1}{2} D_l^{-1} \big(\vec{f}_l - A_l \vec{u}_l\big)$.

- Using a relaxation method to reduce **smooth** error components
- Using a coarse-grid correction (CGC) method to provide a **coarse** approximation
- Key to success: Make smoother and CGC compensate each other

THANKS

Chensong Zhang, AMSS
http://lsec.cc.ac.cn/~zhangcs

Release version 2024.10.07