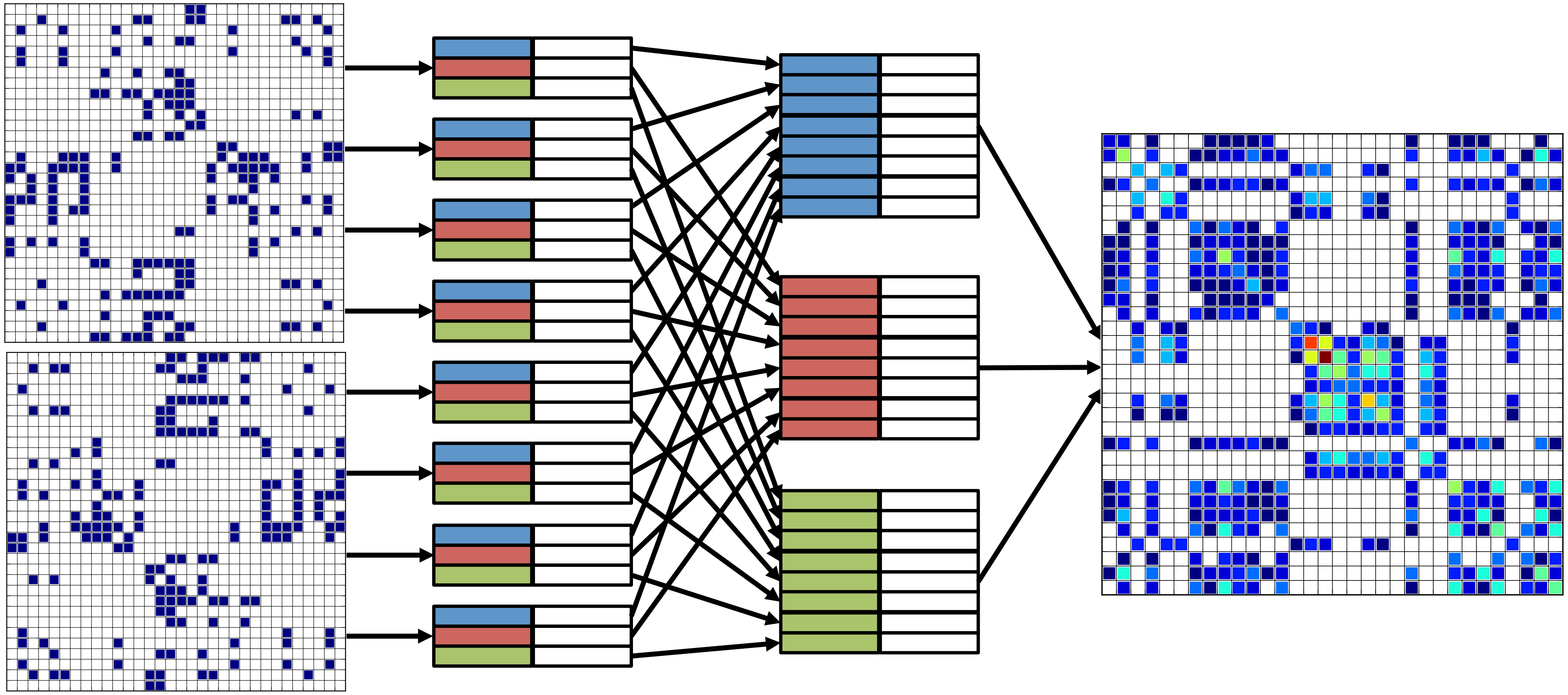
Opportunities in developing a more robust and scalable multigrid solver

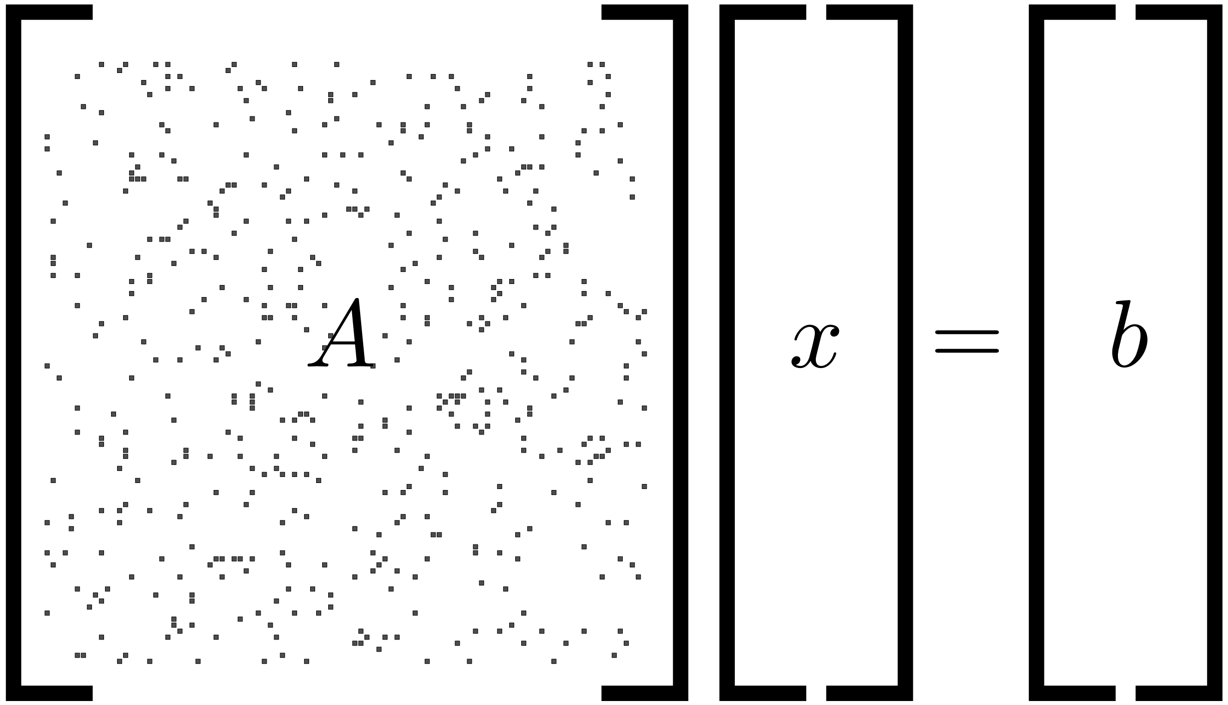
Joint Lab Workshop June 2013

Luke Olson
Department of Computer Science
University of Illinois at Urbana-Champaign



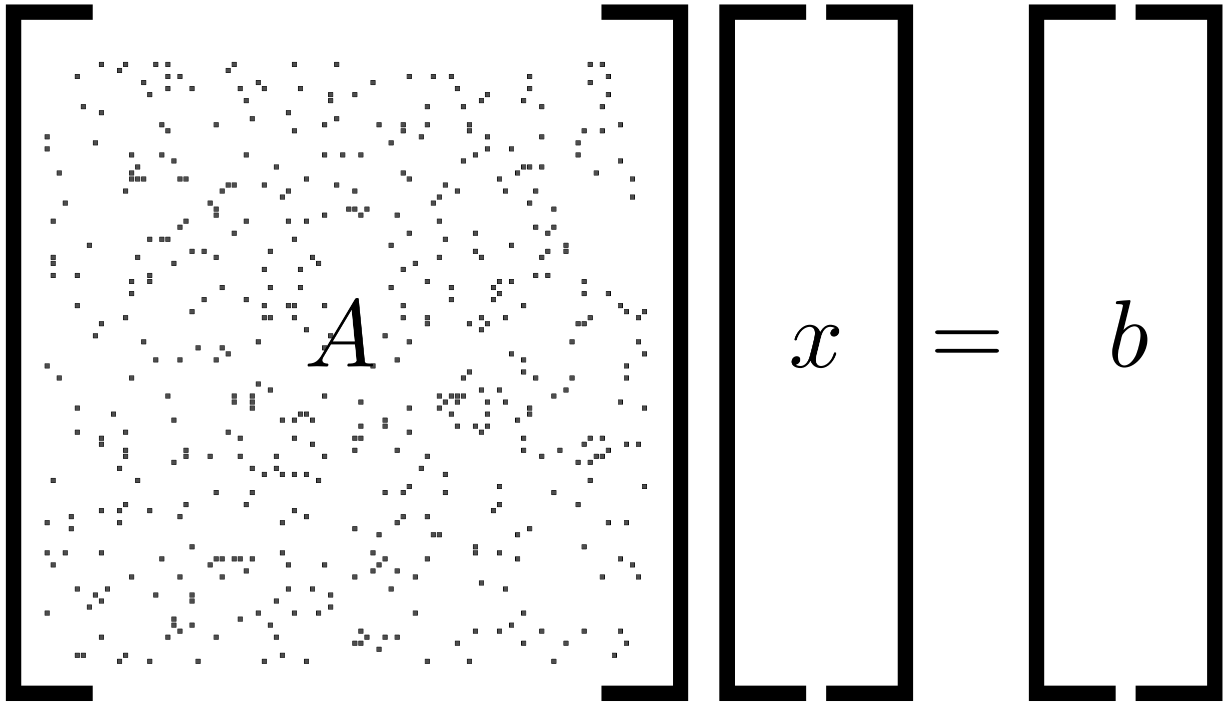
problem

wanted: to solve large-scale, non-elliptic problems



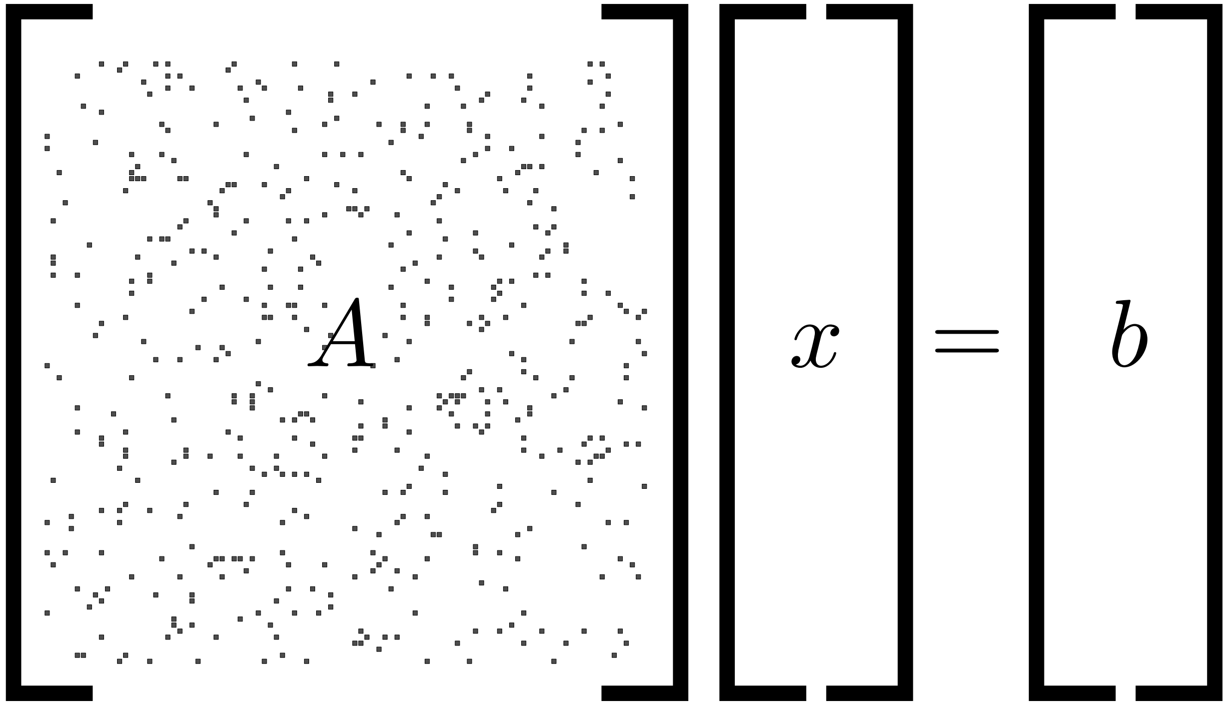
- challenges:
  - complex, non-symmetric, unstructured problems
  - computing environments less homogeneous
    --- e.g. high throughput

solvers challenge (classic approach)



- Focus solvers development on
  - **robustness** --- i.e., improve convergence
  - **scalability** --- i.e., improve weak scaling

solvers challenge (now)



- Focus solvers development on
  - **mapping optimal strategies to software/arch**
  - **utilizing architectural advantages**
  - **software flexibility*****

***see Jed Brown’s talk

The point of this talk

Highlight two advances in multigrid

1. optimal strategies for multigrid robustness
2. performance strategies for multigrid for high-throughput

Identify two challenge areas for collaboration

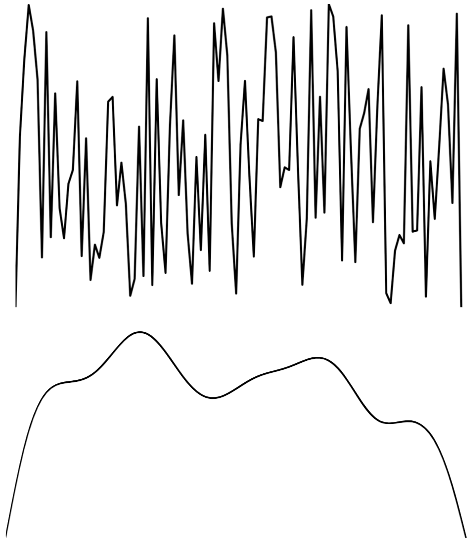
1. bringing optimizations to scale
2. integrating high-throughput advances

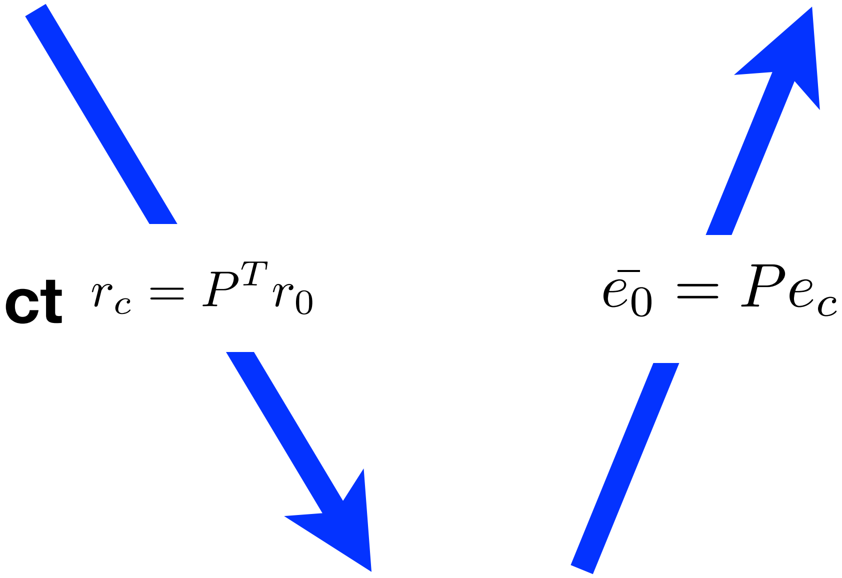
Multilevel view

1

1. attenuate high energy quickly with with relaxation

2. attenuate low energy error through coarse-grid correction





relax $x_0 \leftarrow x_0 + \omega D^{-1} r_0$

$x_0 \leftarrow x_0 + \bar{e_0}$ **correct**

restrict $r_c = P^T r_0$

$\bar{e_0} = P e_c$ **interpolate**

$(P^T A P) e_c = r_c$

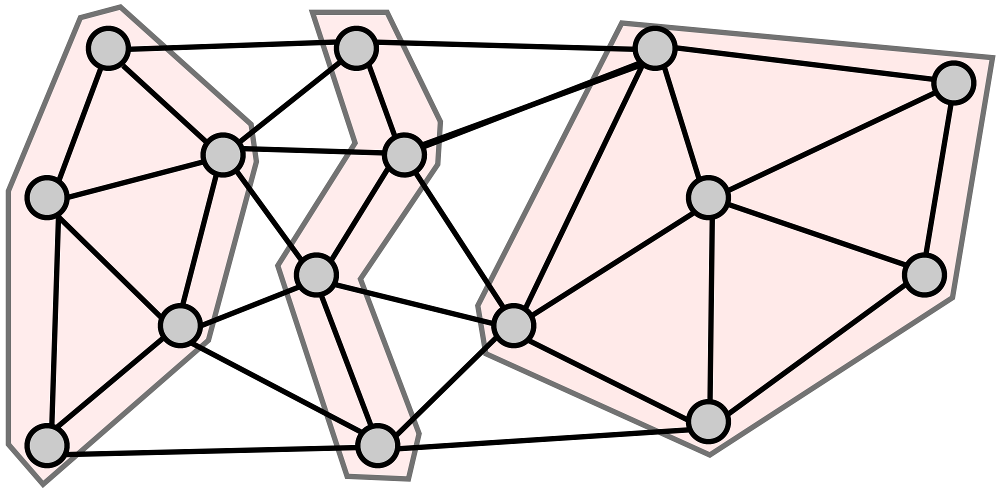
coarse solve

need P

Which multigrid method?

- none. Think of a *framework.*

- example: aggregation groups of fine nodes form coarse nodes

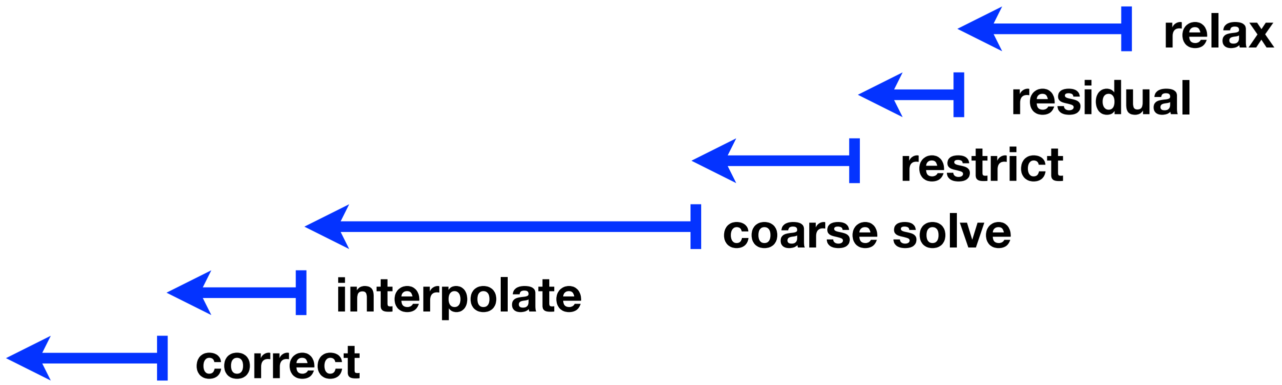


fine: 15

coarse: 3

- this gives a pattern for P

$$e_1 \leftarrow (I - P(P^T AP)^{-1} P^T A)Ge_0$$



relax

residual

restrict

coarse solve

interpolate

correct

Typical Components

1

Setup

find low energy: physics, adaptive methods, intuition

strength measure between d.o.f.:
edge weights, relaxation

coarse point — fine point mapping:
geometric, aggregation, independent set

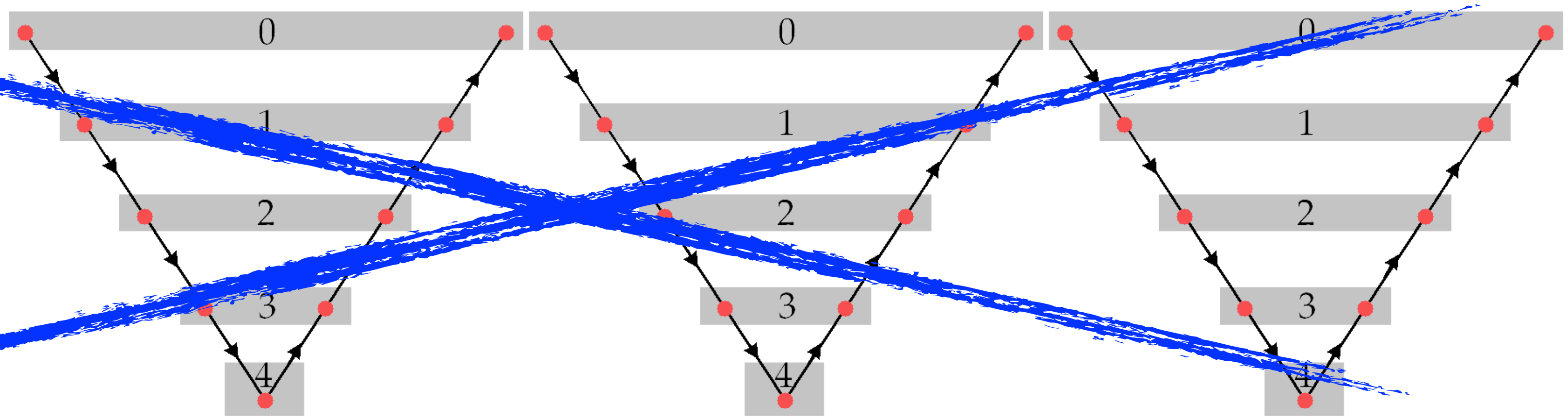
low complexity, accurate interpolation:
weighted averages, relaxation, **energy-minimization**

P

Solve

better cycling

- richer coarse grids



optimizing energy

1

$$e_1 \leftarrow \underbrace{(I - P(P^TAP)^{-1}P^TA)}_{\text{coarse grid correction}}\underbrace{Ge_0}_{\text{relax}} \in \mathcal{R}(P)$$

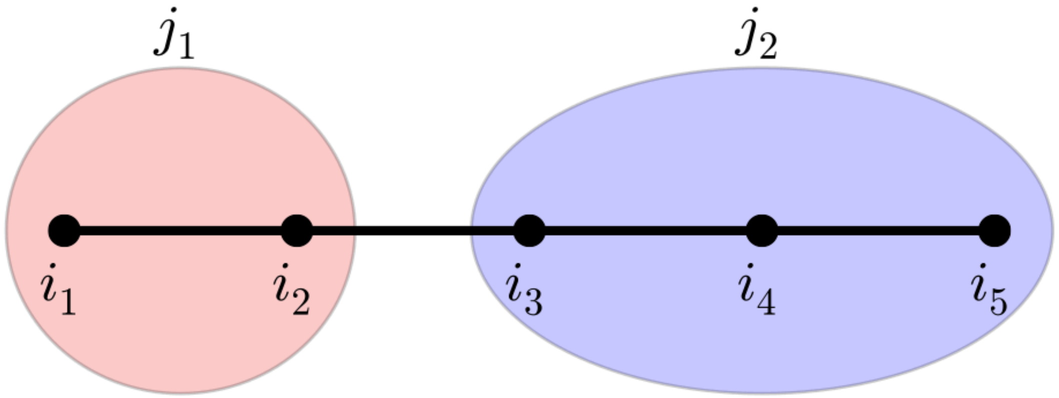
- P should have low energy (low A-norm or A^*A-norm)
  1. determine sparsity pattern
  2. minimize energy column-wise (parallel)

*** Olson, Schroder, Tuminaro, A *general interpolation strategy for algebraic multigrid using energy-minimization, SISC,* 2010.

Interpolation: standard approach

1

- Set the sparsity pattern from aggregation



Text

$$Agg = \begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \end{array} \begin{array}{cc} j_1 & j_2 \\ \hline 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{array}$$

$$P_{tent}$$

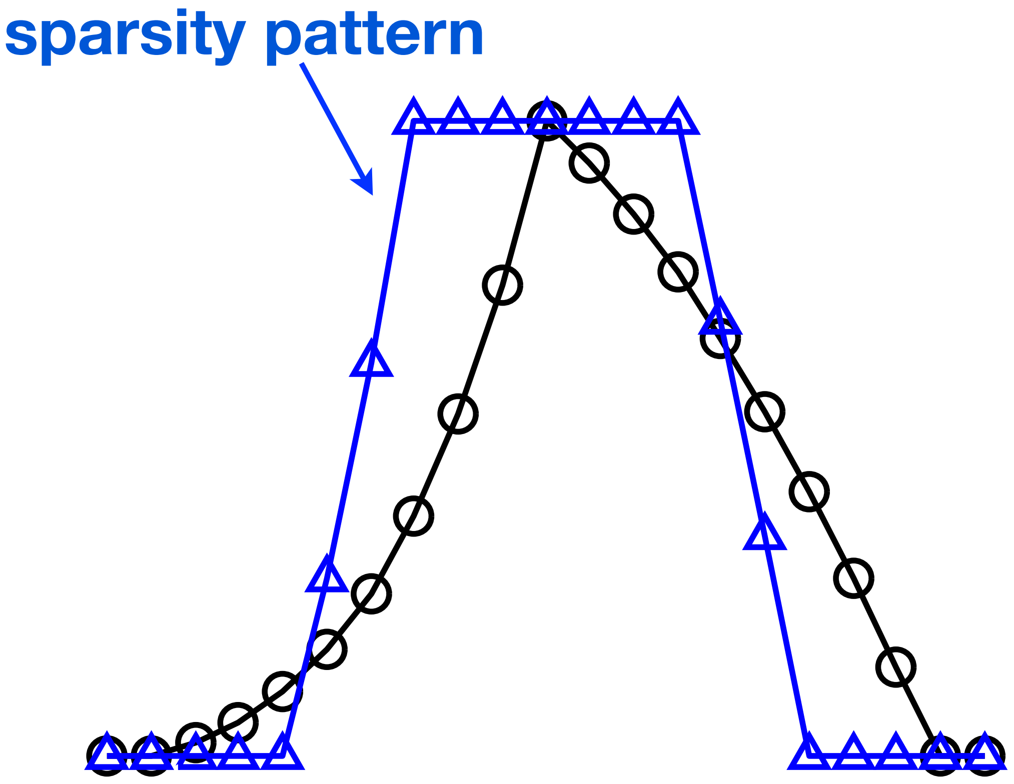
$$P \leftarrow (I - \omega D^{-1} A) P_{tent}$$

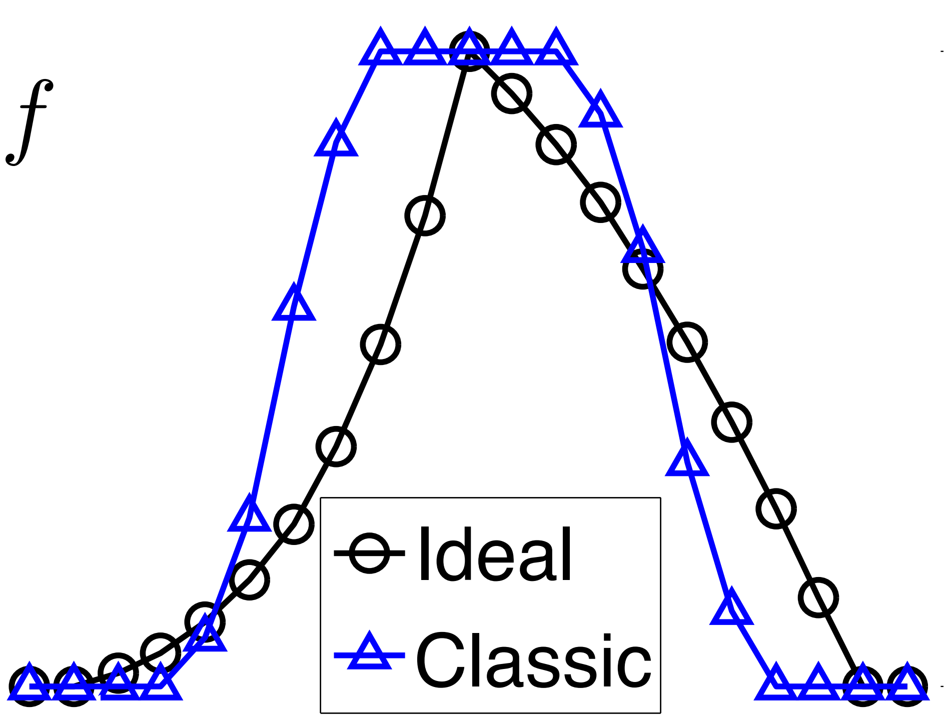
- ● reduce energy
- ● improve accuracy
- ● increase complexity

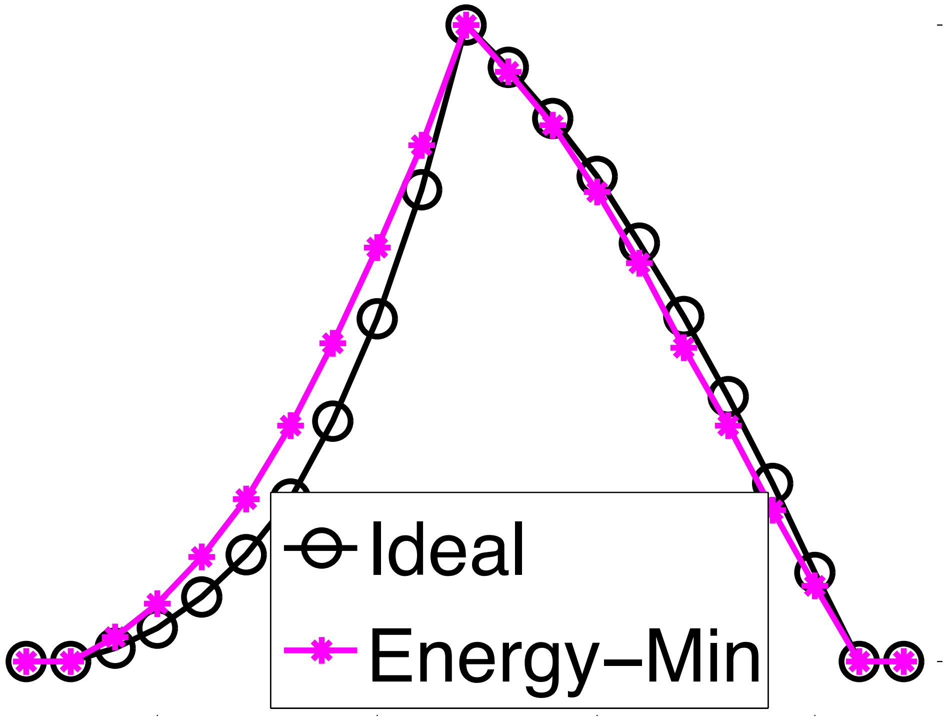
causes dense coarse matrices,
more communication

Range of Interpolation

$$-u_{xx} + \sin(x)u_x = f$$







Toward General Interpolation

1

- Want P so that $u_{low} \in \mathcal{R}(P)$

1. Grow and fix sparsity pattern as $S^k P_{tent}$

strong graph

2. Minimize residual of

$$AP_j = 0 \quad \text{for each column } j$$

3. Constraint the minimization with

$$Pu^c_{low} = u_{low}$$

Toward General Interpolation

1

- Hermitian (and positive definite): use **CG**

$$AP_j = 0 \Leftrightarrow \min \|P_j\|_A$$

$$R = P^*$$

- Non-Hermitian: use **GMRES**

$$AP_j = 0 \Leftrightarrow \min \|P_j\|_{A^*A}$$

$$A^*R_j^* = 0 \Leftrightarrow \min \|R_j^*\|_{AA^*}$$

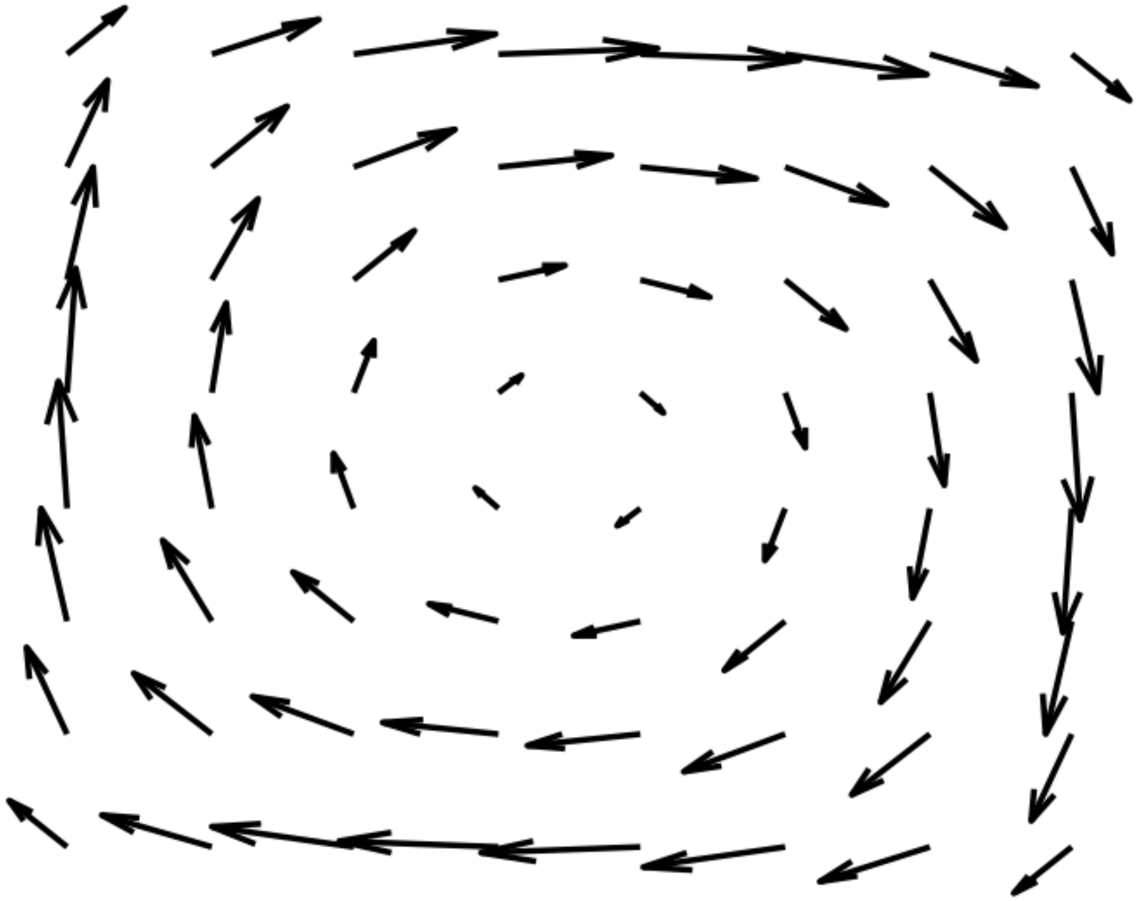
- Range of interpolation targets “right” low-energy

- Range of restriction* targets “left” low-energy

- Cost is comparable to that of standard smoothing

Example: recirculating flow

1

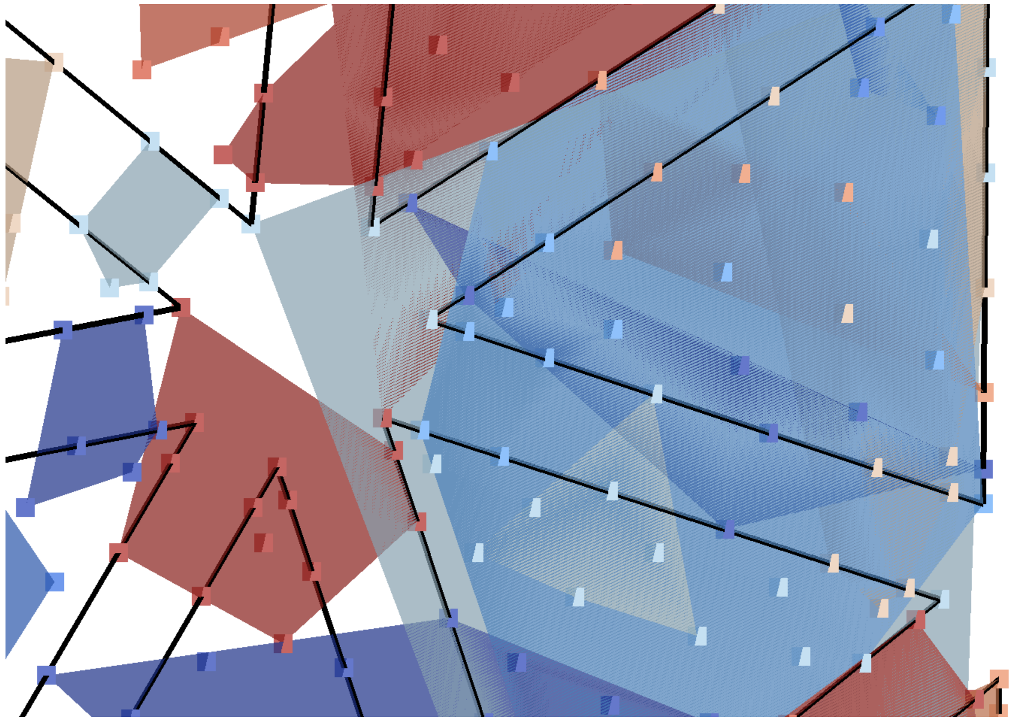


h	std.	opt.
1/64	>150	24
1/128	>150	28
1/256	>150	33
1/512	>150	33

⊢iterations⊣

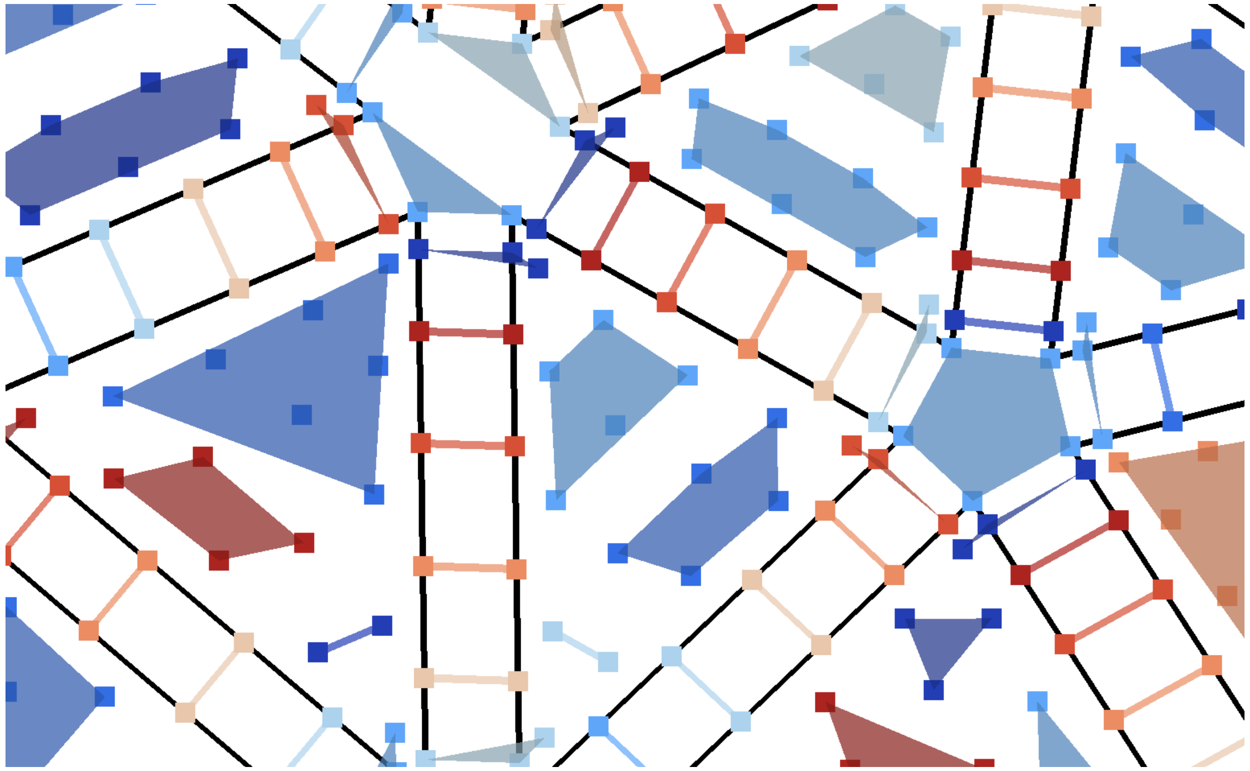
High-order discontinuous Galerkin: diffusion

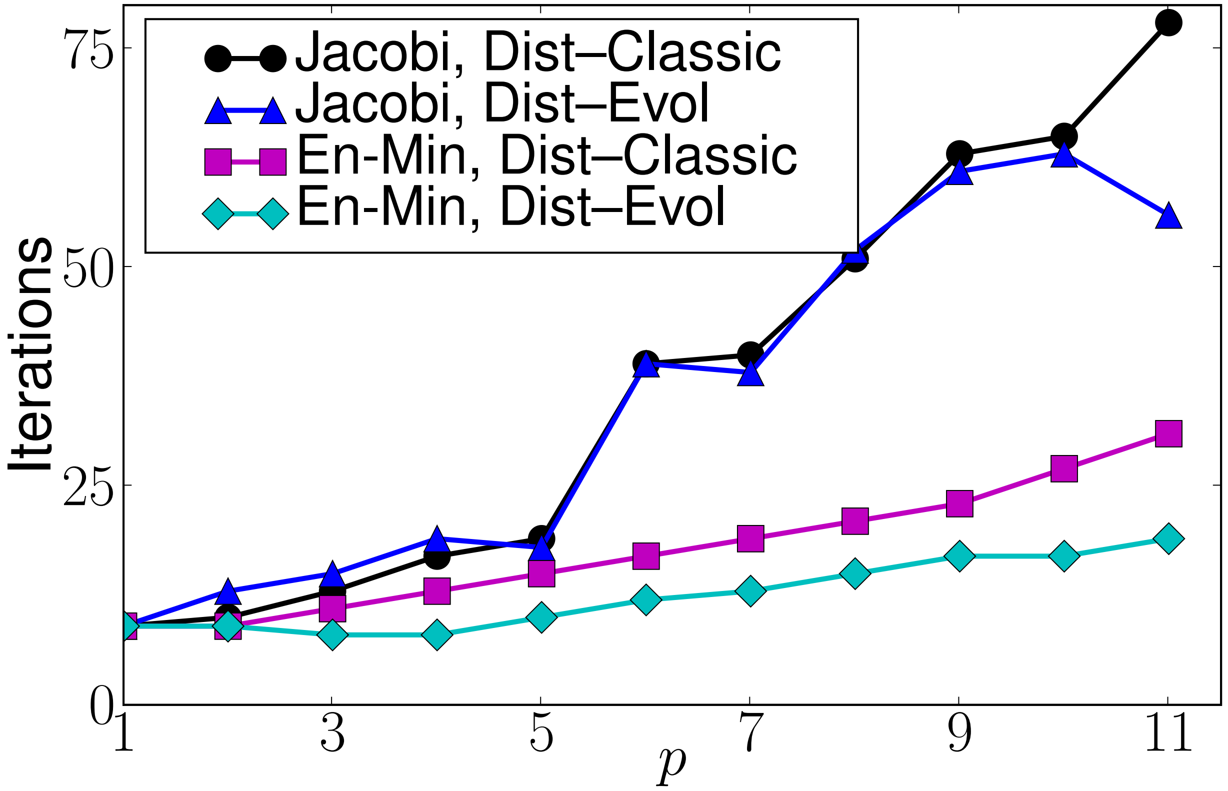
1



key ingredients:

- conforming aggregations step
- adapt the near null space
- optimal interpolation





Collaboration #1

1

Opportunities

I. Optimal interpolation at scale

- a. many decisions:
  - optimize on communication distance, size, impact
  - local vs non-local optimizations
- b. geometric-style optimization
- c. on-the-fly updates to the hierarchy (time, nonlinear, etc)
- d. DD

II. Other optimizations:

- a. adaptive setup
- b. dynamic aggregation

Why Acceleration for Multigrid

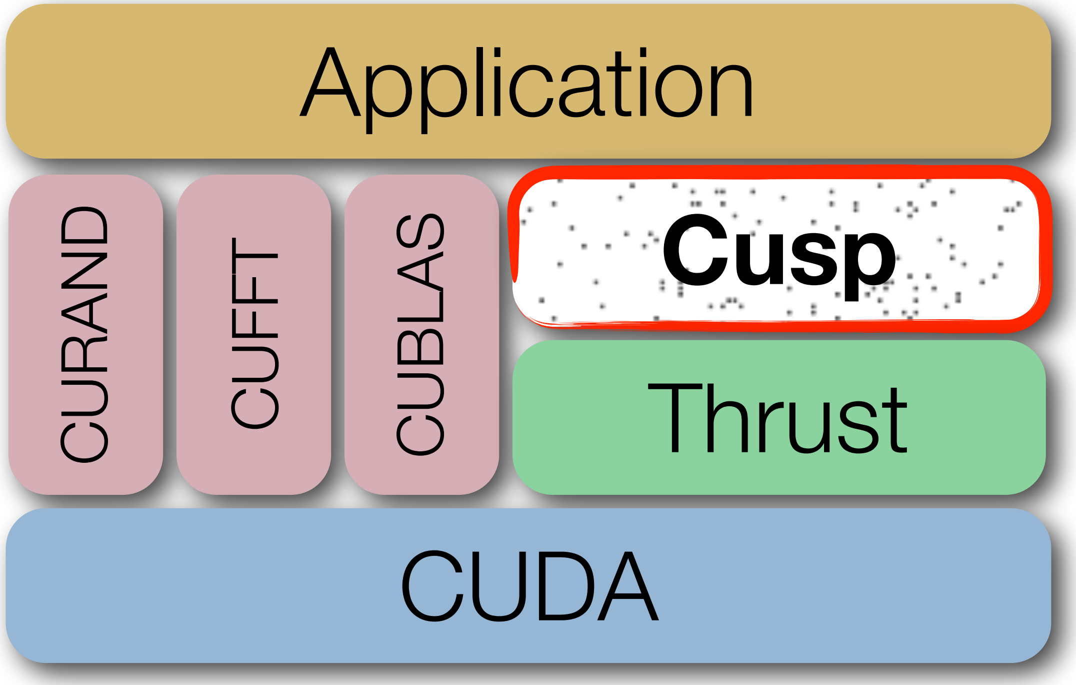
potential: SpMVs*** are fast,
scans+reductions are fast

useable software: CUDA + Thrust + Cusp

AMG "asks" for acceleration:
✓adaptive
✓thick near null-space
✓higher intensity work in optimizations

*** see Bill Gropp's talk

CUSP

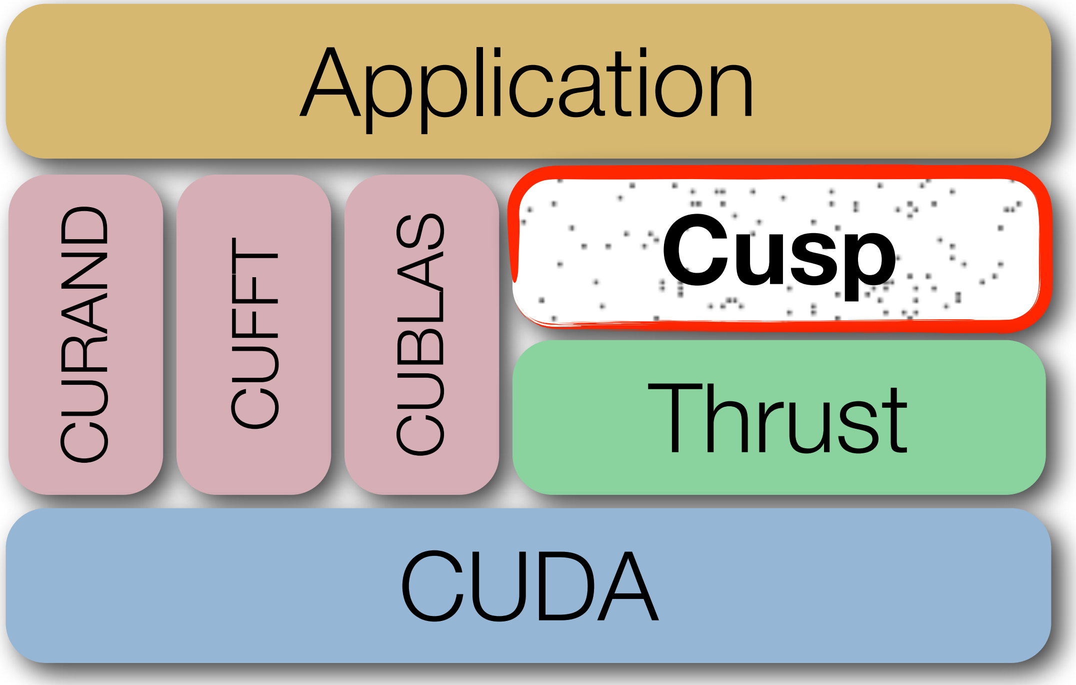


- expose fine-grained parallelism
- utilize fast kernels (gather, scatter, scans, sort, etc)

*** Bell, Dalton, Olson, *Exposing fine-grained parallelism in algebraic multigrid, 2011*.

CUSP

- fast development
- low overhead
- open source



•expose fine-grained parallelism
•utilize fast kernels (gather, scatter, scans, sort, etc)

*** Bell, Dalton, Olson, *Exposing fine-grained parallelism in algebraic multigrid, 2011*.

SpMM

- SMMP algorithm: very sequential
  - requires O(ncol) storage to determine entries of each sparse row
  - parallelism would require O(ncol) memory per thread
- Consider $C = A * B$

$$A = \begin{bmatrix} 5 & 10 & 0 \\ 15 & 0 & 20 \end{bmatrix}, = \begin{bmatrix} (0,0,\ \ 5) \\ (0,1,10) \\ (1,0,15) \\ (1,2,20) \end{bmatrix}, \quad B = \begin{bmatrix} 25 & 0 & 30 \\ 0 & 35 & 40 \\ 45 & 0 & 50 \end{bmatrix}, = \begin{bmatrix} (0,0,25) \\ (0,2,30) \\ (1,1,35) \\ (1,2,40) \\ (2,0,45) \\ (2,2,50) \end{bmatrix},$$

1. form intermediate view of C
2. sort C by row, col
3. contract C by summing duplicates

SpMM

$$A = \begin{bmatrix} 5 & 10 & 0 \\ 15 & 0 & 20 \end{bmatrix}, B = \begin{bmatrix} 25 & 0 & 30 \\ 0 & 35 & 40 \\ 45 & 0 & 50 \end{bmatrix},$$

$\boxed{2}$

- **Expand Primitives:**
  `reduce, scatter, scan`
  expand with $A(i,j) * B(i,:)$

$$C = \begin{bmatrix} (0,0,\ \ 125) \\ (0,2,\ \ 150) \\ (0,1,\ \ 350) \\ (0,2,\ \ 400) \\ (1,0,\ \ 375) \\ (1,2,\ \ 450) \\ (1,0,\ \ 900) \\ (1,2,1000) \end{bmatrix}$$

- **Sort Primitives:**
  sort by column keys

$$C = \begin{bmatrix} (0,0,\ \ 125) \\ (0,1,\ \ 350) \\ (0,2,\ \ 150) \\ (0,2,\ \ 400) \\ (1,0,\ \ 375) \\ (1,0,\ \ 900) \\ (1,2,\ \ 450) \\ (1,2,1000) \end{bmatrix}$$

SpMM

$\boxed{2}$

- **Contract Primitives:** `reduce`

$$C = \begin{bmatrix} (0,0,\ \ 125) \\ (0,1,\ \ 350) \\ (0,2,\ \ 550) \\ (1,0,1275) \\ (1,2,1450) \end{bmatrix} = \begin{bmatrix} 125 & 350 & 550 \\ 1275 & 0 & 1450 \end{bmatrix}.$$

- insensitive to irregularity of input

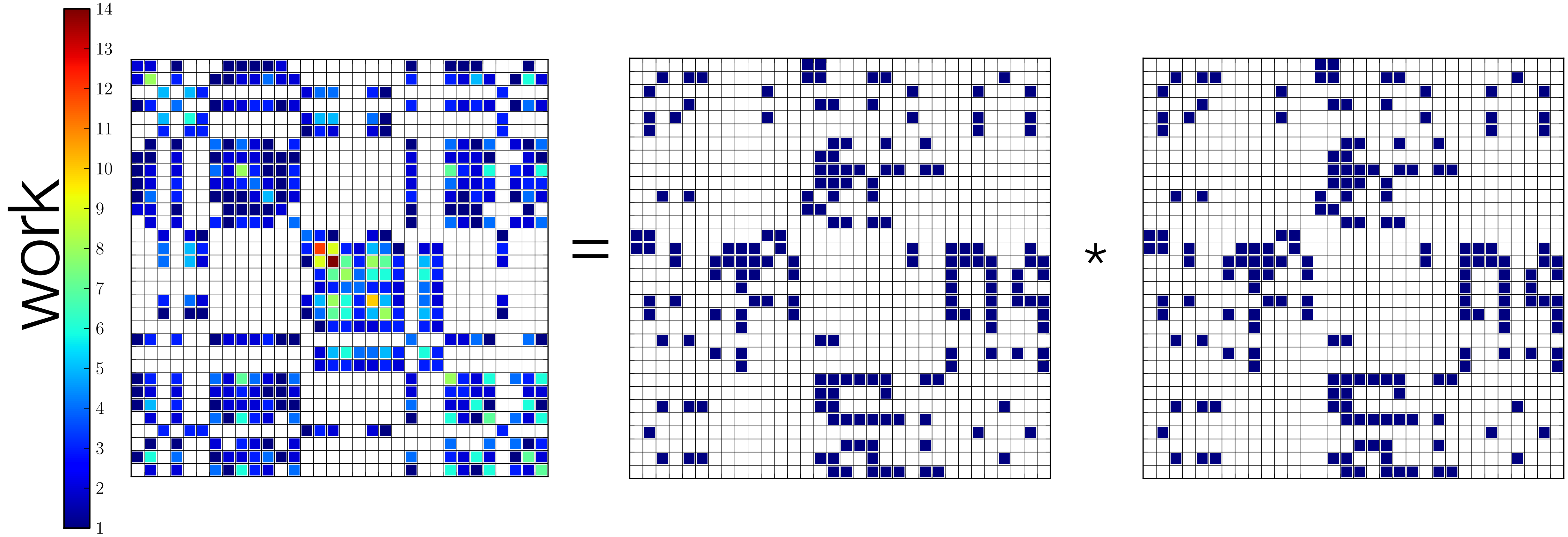
- same “work” as SMMP

- storage cost can be large for intermediate (reduce by subdividing)

SpMM Modeling $C = A * B$

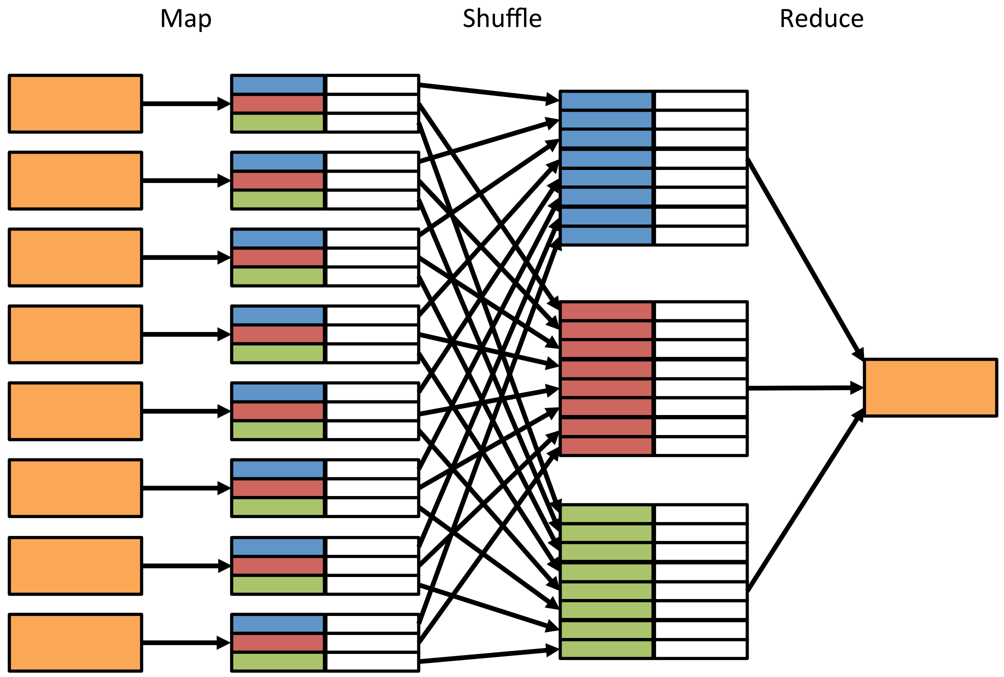
- Structure of C expensive to (accurately) ascertain

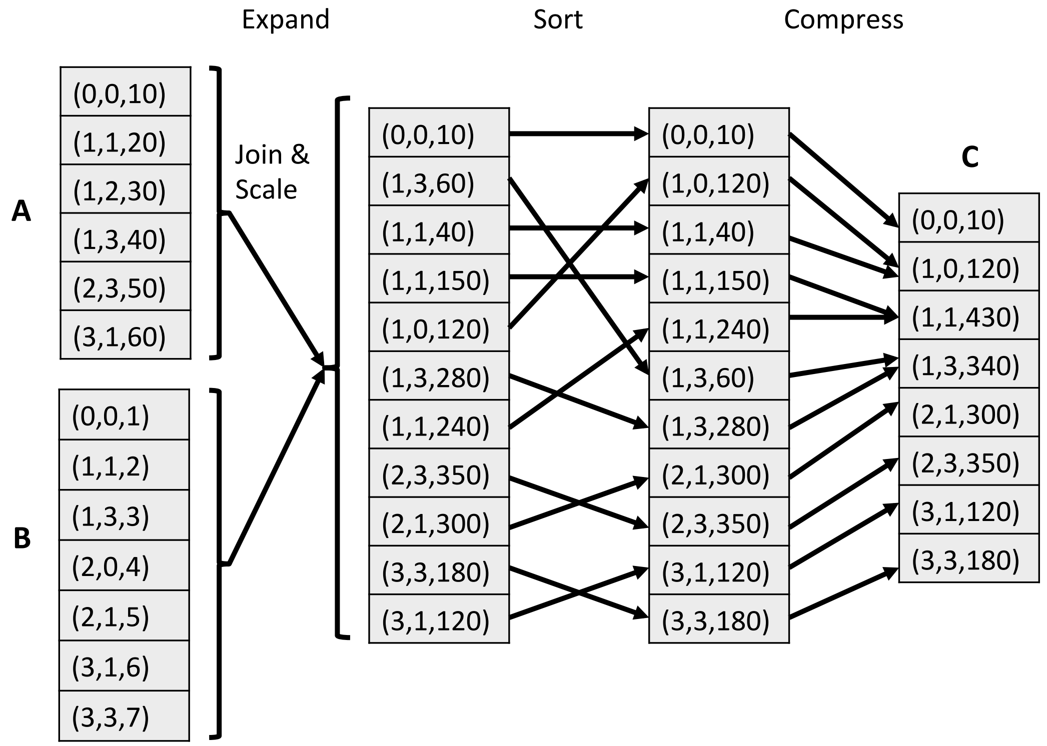
- Structure of C not representative of work



SpMM Modeling

2





A

(0,0,10)
(1,1,20)
(1,2,30)
(1,3,40)
(2,3,50)
(3,1,60)

B

(0,0,1)
(1,1,2)
(1,3,3)
(2,0,4)
(2,1,5)
(3,1,6)
(3,3,7)

Expand

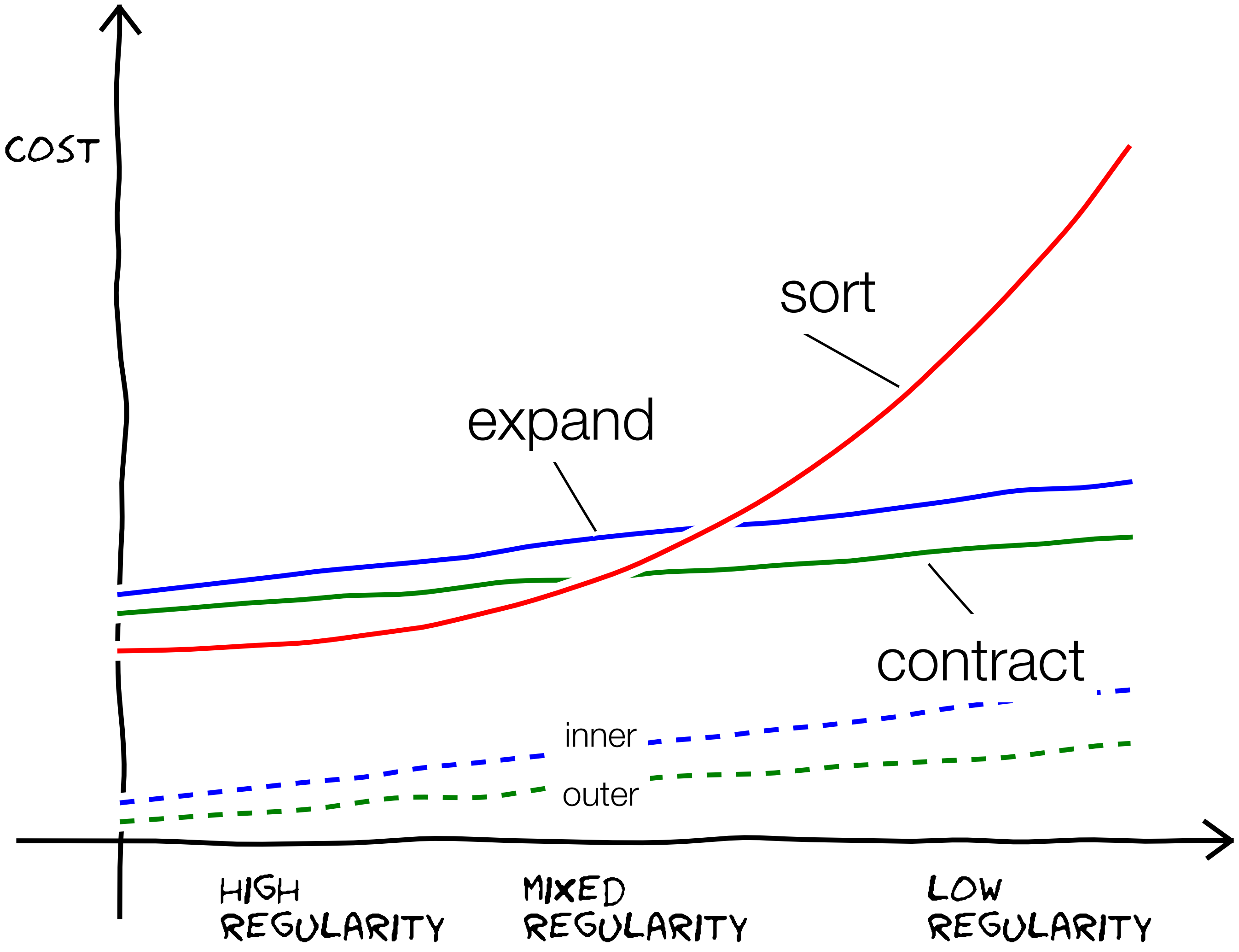
(0,0,10)
(1,3,60)
(1,1,40)
(1,1,150)
(1,0,120)
(1,3,280)
(1,1,240)
(2,3,350)
(2,1,300)
(3,3,180)
(3,1,120)

Sort

(0,0,10)
(1,0,120)
(1,1,40)
(1,1,150)
(1,1,240)
(1,3,60)
(1,3,280)
(2,1,300)
(2,3,350)
(3,1,120)
(3,3,180)

C

(0,0,10)
(1,0,120)
(1,1,430)
(1,3,340)
(2,1,300)
(2,3,350)
(3,1,120)
(3,3,180)



Collaboration #2

2

Opportunities

I. SpMM and other non-linear algebra optimized linear algebra optimizations

a. paraphrase Gropp: *not everything should be reduced to linear algebra*

b. How to use in a multinode-multiGPU environment?

II. Can we use hardware optimized scans/reduces at scale?

a. other programming models support this

b. P. Fischer makes at good case at CSE13***

III.How to incorporate low-level (useable) abstractions

a. CUSP flexible back-end

b. Better way to use, manage back-ends in a library code

c. DD?

***P. Fischer, PDE-Based Simulation Beyond
Petascale, CSE13

Summary of potential collaboration:

1. **Redevelop optimized multigrid components in a large-scale environment**

2. **Integrate architecture motivated multigrid decisions into a heterogeneous environment**

A comment on future collaboration:

3. **Outline a path or roadmap or position on resilience in solvers**

Looking ahead to more collaboration

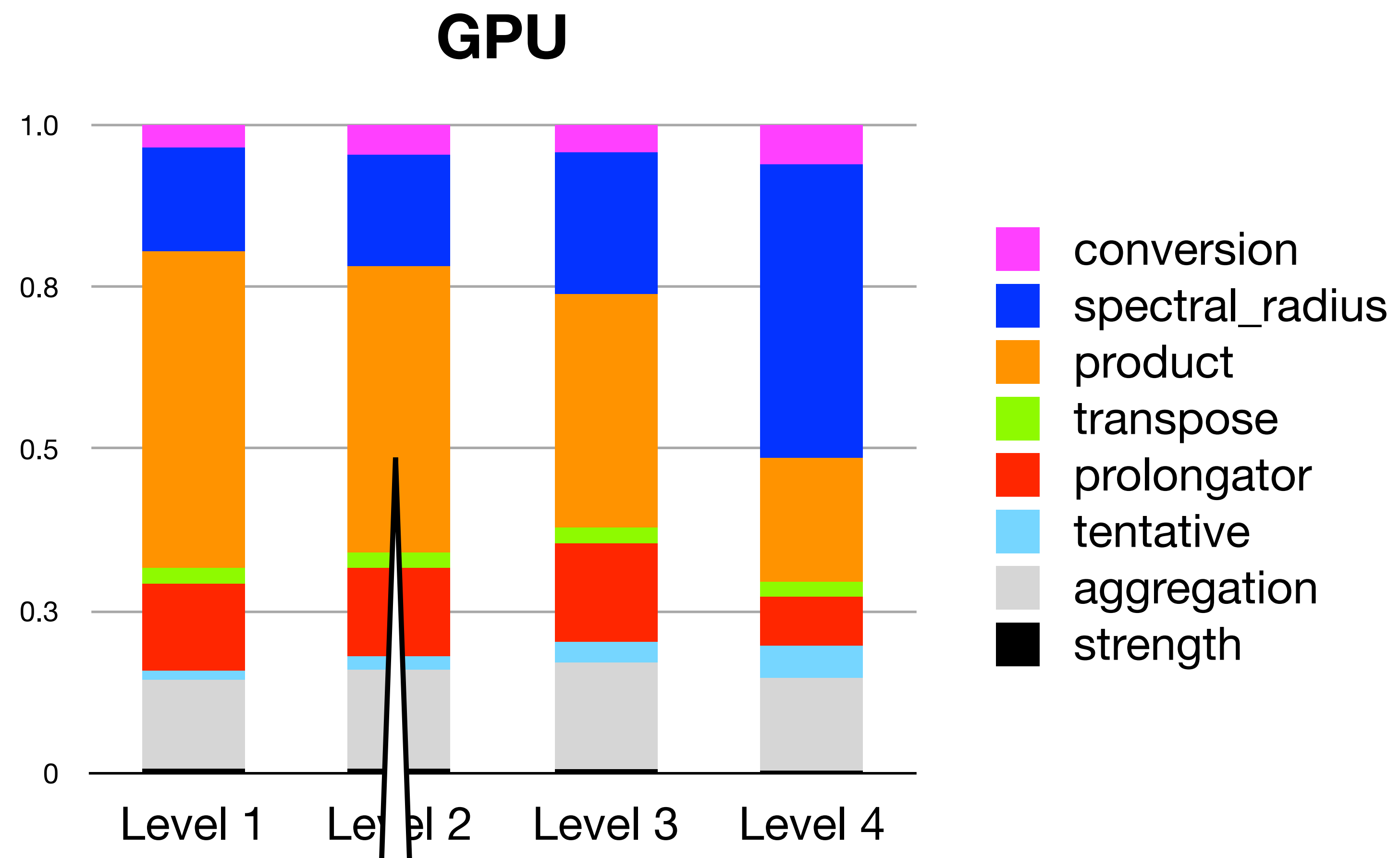
- Students can be a great conduit for moving forward
  - one plan:
    - student from Illinois 1/2 at ANL 1/2 in France for the summer
      + a shorter visit to France during Winter Break
  - adjoint plan:
    - student from France 1/2 at ANL 1/2 at Illinois for the summer
      + a short visit to Illinois (!) during Winter Break

- Co-developing a code
  - Take something like GAMG as a base and fork it
  - Trying this currently with ANL
  - Retains buy-in to a code “structure”, but not a framework
  - Allows ownership for a researcher or student or whomever

- Need a specific plan to carry out over the next 6 mo

- Nvidia for hardware
- software development: CUSP::MG
- software development: PyAMG
- LLNL, SNL for student support

GPU



Triple products are expensive