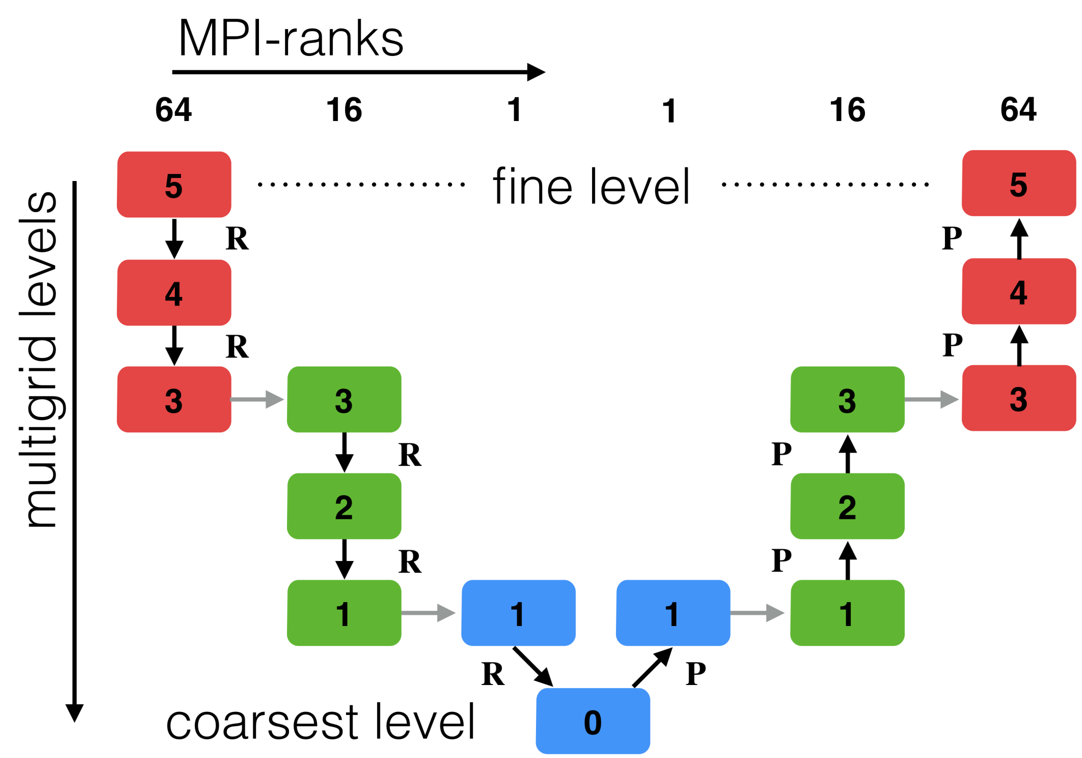
Extreme-scale Multigrid Components within PETSc



Dave A. May (ETH Zürich), **Patrick Sanan** (USI Lugano, ETH Zürich), **Karl Rupp**, **Matthew G. Knepley** (Rice University), **Barry F. Smith** (Argonne National Lab)

Outline

1. **Motivation** : The need for (easy-to-use) agglomeration within extreme-scale geometric multigrid

2. **Implementation** :
    1. The `PCTelescope` implementation
    2. Use cases

3. **Numerical Experiments**

4. **Future Development** : Extensions for unstructured grids

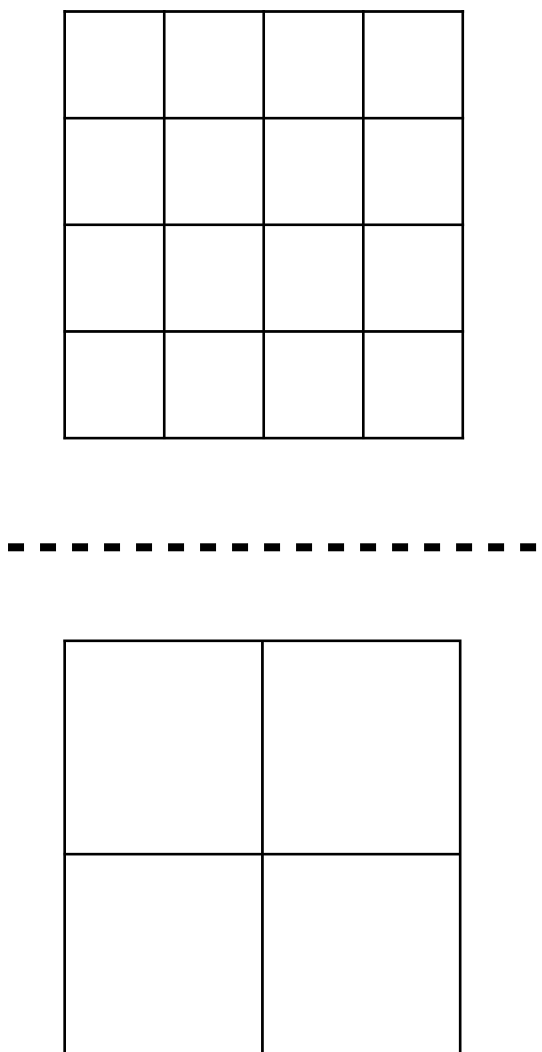
The Need for Agglomeration in Parallel Multigrid

Re-discretised Geometric Multigrid (RMG)

Given $Ax = b$ let v denote our guess for x

Two-level RMG algorithm (Simplest Form)

FINE Ω^h



COARSE Ω^{2h}

"Smooth" N times

$$v = v + \omega(b - Av)$$

Compute residual

$$r^h = b - Av$$

Restrict $r^h \to \Omega^{2h}$, yielding r^{2h}

Compute error

Solve $A^{2h}e^{2h} = r^{2h}$

Interpolate $e^{2h} \to \Omega^h$, yielding e^h

Compute residual correction

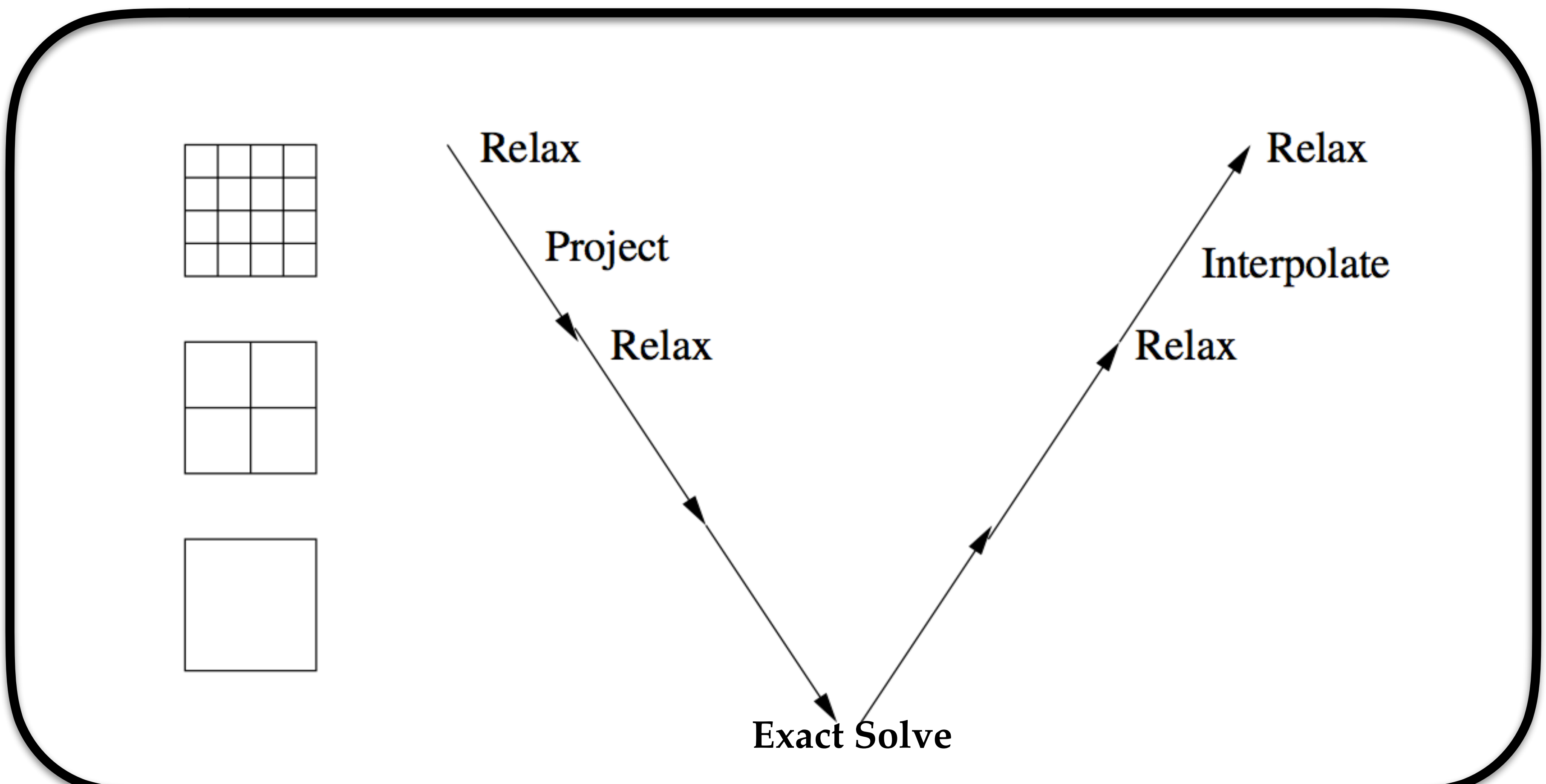
$$v = v + e^h$$

"Smooth" N times

$$v = v + \omega(b - Av)$$

Re-discretised Geometric Multigrid (RMG)

- Ingredients
  - A mesh hierarchy (fine → coarse) on which will discretise our PDE
  - A restriction operator (maps field from fine → coarse)
  - An interpolation operator (maps field from coarse → fine)
  - A smoother on each level
  - A coarse grid solver



Why Multigrid (MG)?

- Theoretically optimal solve time *O(n)* ⟶ **scalable**

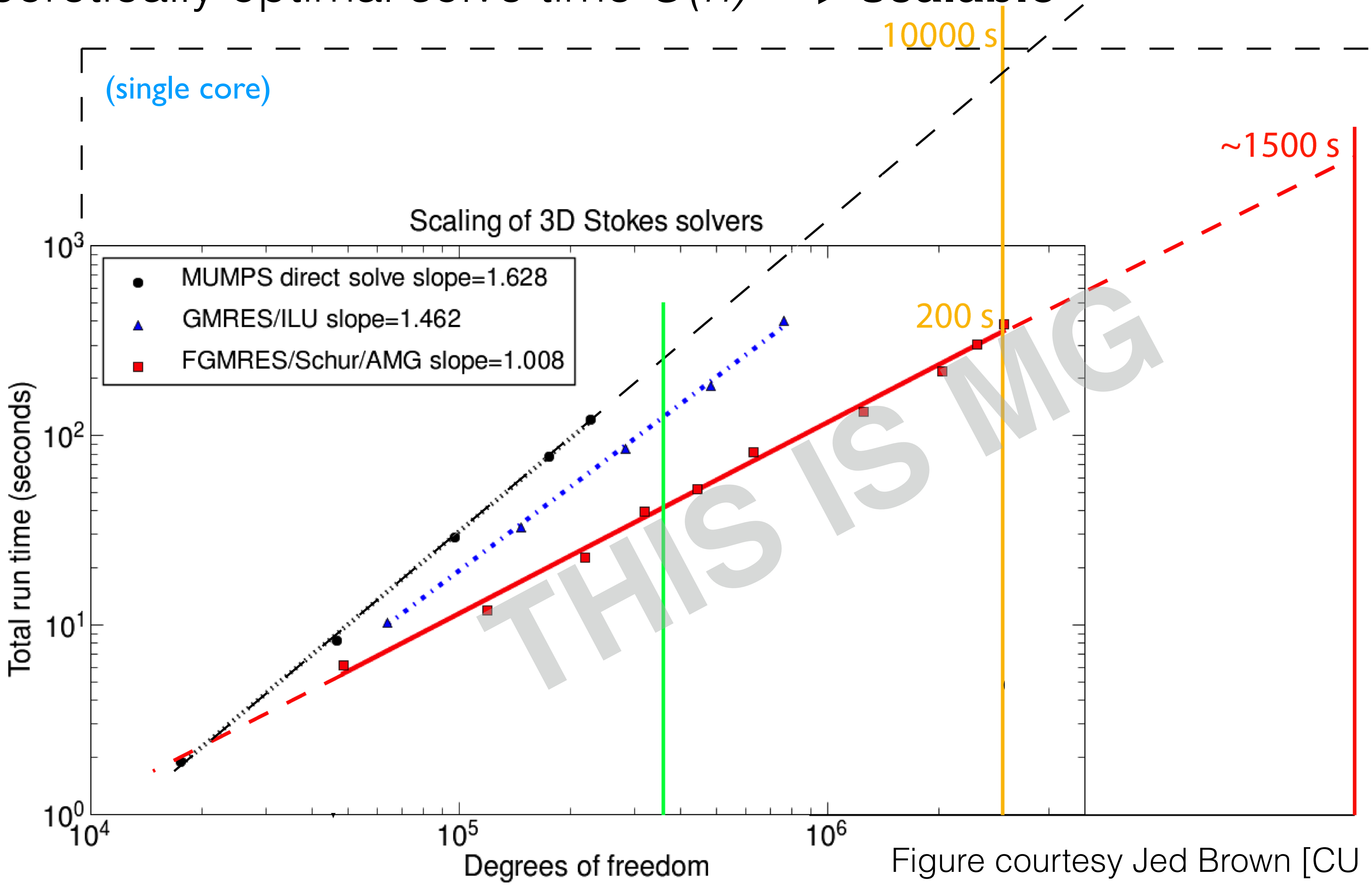


Figure courtesy Jed Brown [CU Boulder]

3000 km x 2000 km x 200 km - 3 velocity components

with mesh resolution of 11 km --> 110 million unknowns

with mesh resolution of 20 km --> 1.8 million unknowns

with mesh resolution of 34 km --> 0.34 million unknowns

Why Are More Levels Better?

- Fewer levels implies the coarse grid will contain a large number of unknowns. Recall that the coarse grid correction requires an accurate solve (usually expensive, non-scalable).
- *Optimality of MG comes from having a tiny coarse grid problem.*

Why Are More Levels Better?

- Fewer levels implies the coarse grid will contain a large number of unknowns. Recall that the coarse grid correction requires an accurate solve (usually expensive, non-scalable).
- *Optimality of MG comes from having a tiny coarse grid problem.*

Two-level method

Mesh	Time	Factor
17^3	1.22E-02	-
33^3	1.25E-01	10x
65^3	3.87E+00	31x
129^3	1.42E+02	63x

Mesh	Levels	Time	Factor
17^3	2	1.22E-02	-
33^3	3	7.14E-02	6x
65^3	4	6.51E-01	9x
129^3	5	5.48E+00	8x
257^3	6	4.37E+01	8x

`$PETSC_DIR/src/ksp/ksp/examples/tutorials/ex45.c`

- *When RMG breaks down and the effective coarsest grid is not "coarse enough" ➔ change to another scalable method*

Why Are More Levels Better?

- Fewer levels implies the coarse grid will contain a large number of unknowns. Recall that the coarse grid correction requires an accurate solve (usually expensive, non-scalable).
- *Optimality of MG comes from having a tiny coarse grid problem.*

Two-level method

Mesh	Time	Factor
17^3	1.22E-02	-
33^3	1.25E-01	10x
65^3	3.87E+00	31x
129^3	1.42E+02	63x

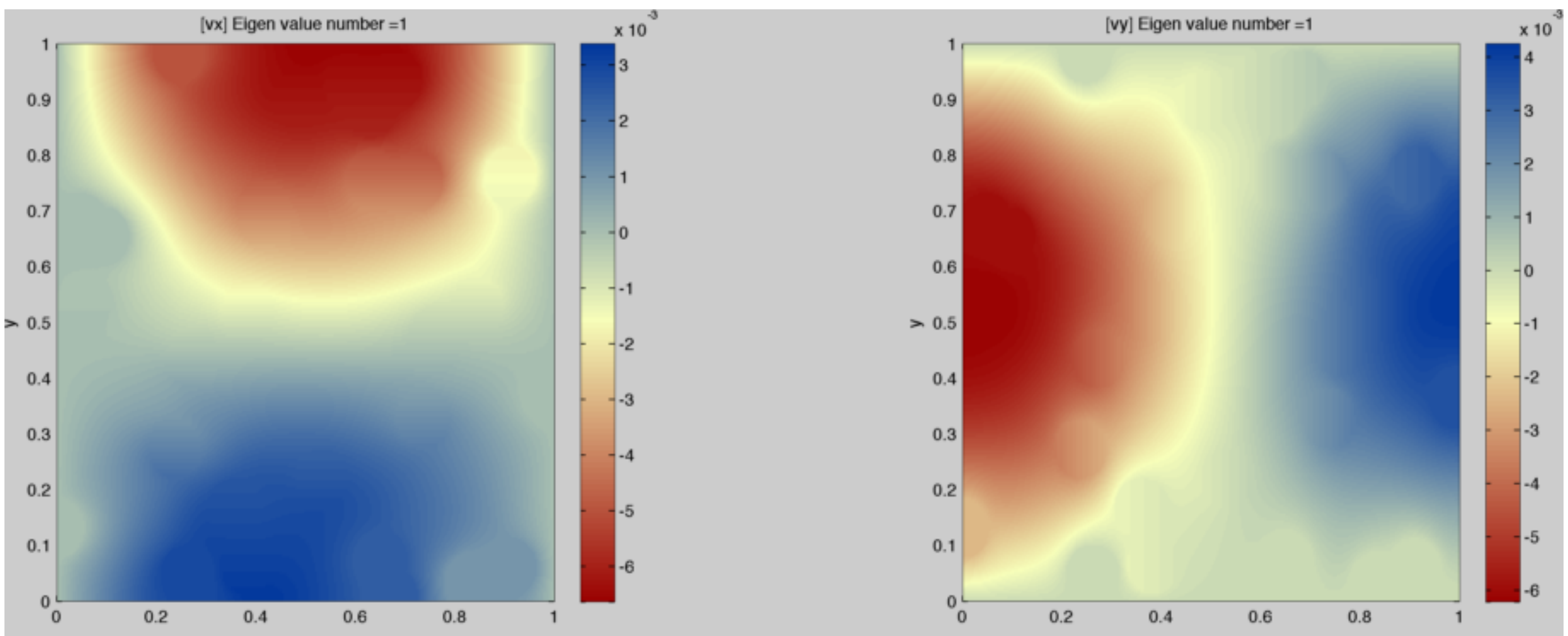
Mesh	Levels	Time	Factor
17^3	2	1.22E-02	-
33^3	3	7.14E-02	6x
65^3	4	6.51E-01	9x
129^3	5	5.48E+00	8x
257^3	6	4.37E+01	8x

`$PETSC_DIR/src/ksp/ksp/examples/tutorials/ex45.c`

- *When RMG breaks down and the effective coarsest grid is not "coarse enough" → change to another scalable method*

Multigrid in Parallel

- Multigrid levels are sometimes limited in practice
  - Practical restrictions often apply; e.g. a minimum of 1 finite element per rank
  - "Empty" ranks may still impose collective communication costs
  - Coarse grids may be limited in their ability to resolve features



- A multigrid V-cycle (with an exact coarse grid solve) is all-to-all communication, with a **fundamental *log(P)* communication cost**

Multigrid in Parallel - Where to Communicate?

- Should the cost be incurred within a solve on a coarse grid?
  - AMG or another multilevel method (e.g. `PCGAMG`)
    - Setup stage doesn't scale for AMG
    - Shifts the question but doesn't fundamentally answer it
  - Hierarchical Krylov methods [May et. al CMAME 2015]
    - Doesn't scale forever - network latency eventually dominates
  - Redundant solve on all cores (PETSc's `PCREDUNDANT`)
    - Slow and expensive
- Or at intermediate points in the hierarchy? ⟶ **Agglomeration**
  - **As we coarsen, use smaller sets of processors (MPI ranks)**
  - Allows balance of communication and computation
  - Well-known, but requires implementation effort

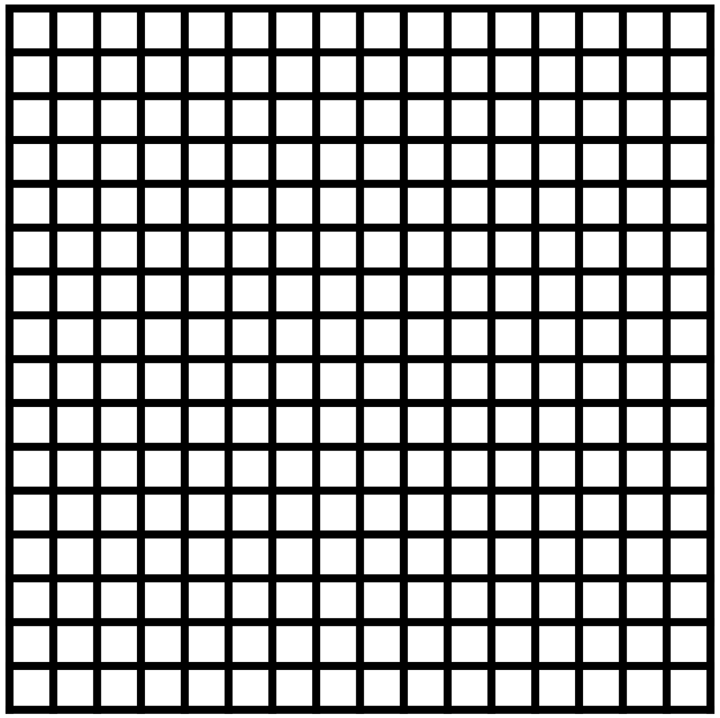
Multigrid in Parallel - Where to Communicate?

- Should the cost be incurred within a solve on a coarse grid?
  - AMG or another multilevel method (e.g. `PCGAMG`)
    - Setup stage doesn’t scale for AMG
    - Shifts the question but doesn’t fundamentally answer it
  - Hierarchical Krylov methods [May et. al CMAME 2015]
    - Doesn’t scale forever - network latency eventually dominates
  - Redundant solve on all cores (PETSc’s `PCREDUNDANT`)
    - Slow and expensive
- Or at intermediate points in the hierarchy? ⟶ **Agglomeration**
  - **As we coarsen, use smaller sets of processors (MPI ranks)**
  - Allows balance of communication and computation
  - Well-known, but requires implementation effort

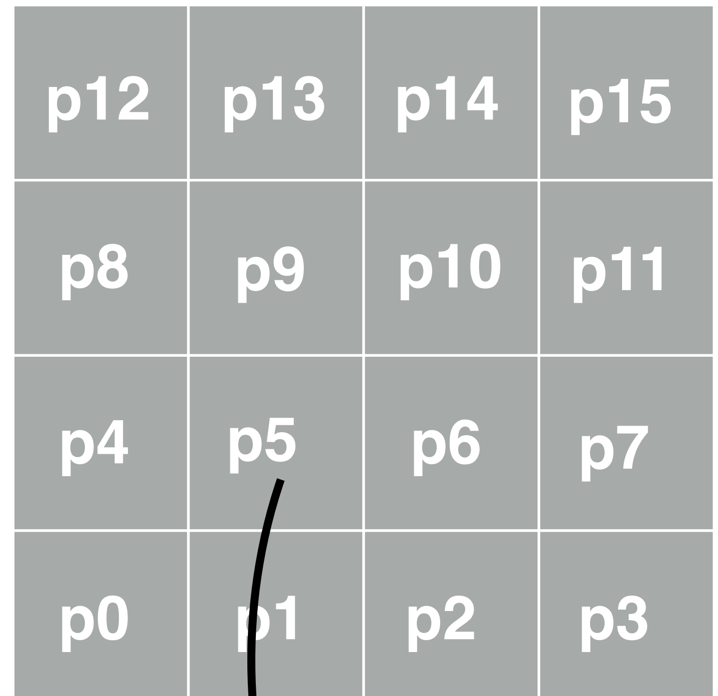
`PCBDDC`
`PCGAMG`

Repartitioning Coarse Grids

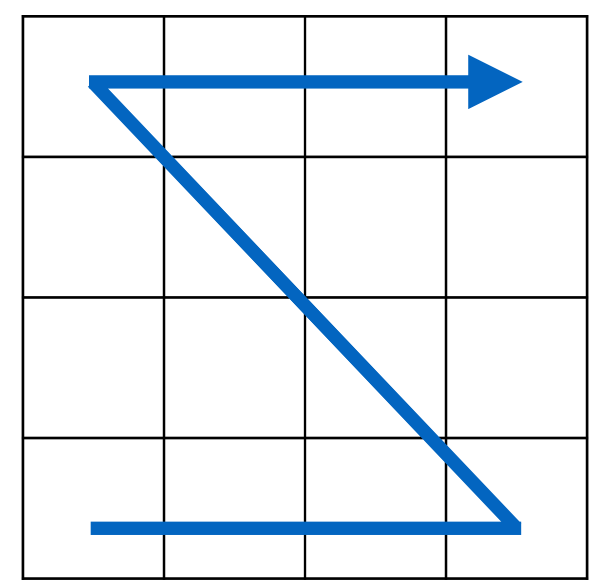
Mesh: 16 x 16 elements



Partition: 4 x 4 processors



Local element ordering (p5)



We wish to solve

$$A^{16}x^{16} = b^{16}$$

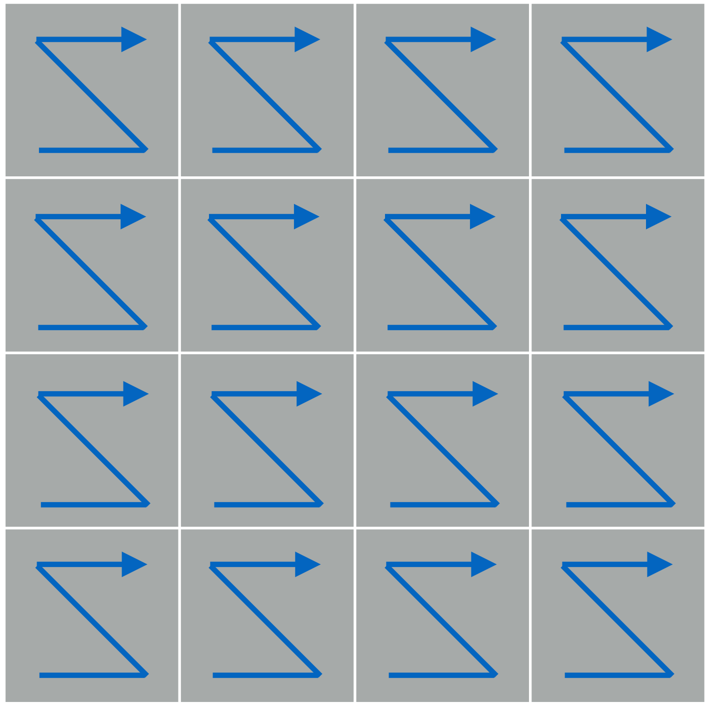
on a smaller number of processors

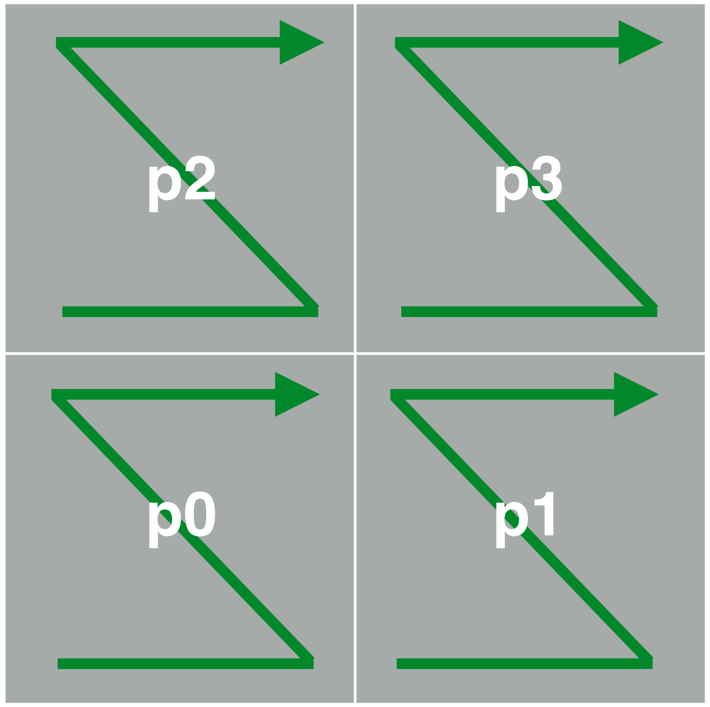
Repartitioning Coarse Grids

Repartition: 2 x 2 processors

A^{16}

b^{16}





A^4

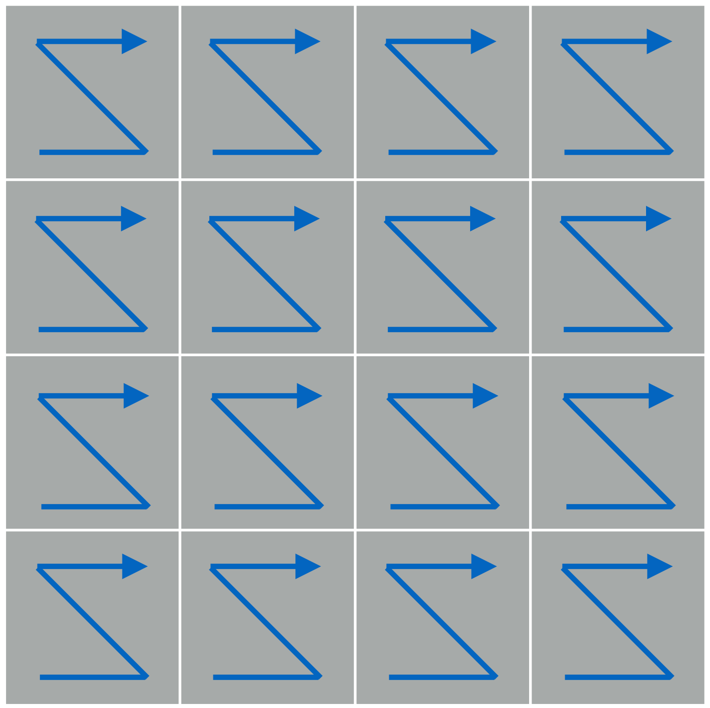
b^4

x^4

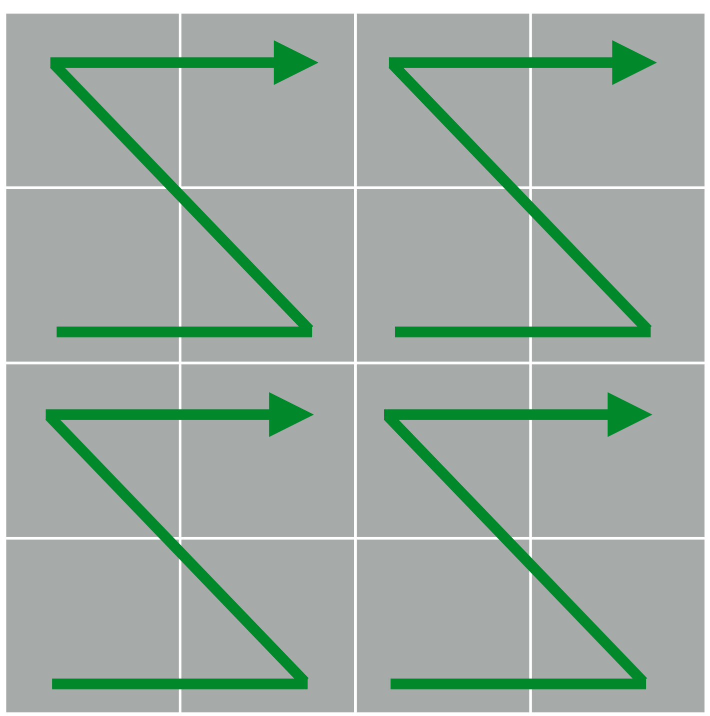
Repartitioning Coarse Grids

A^{16}

b^{16}



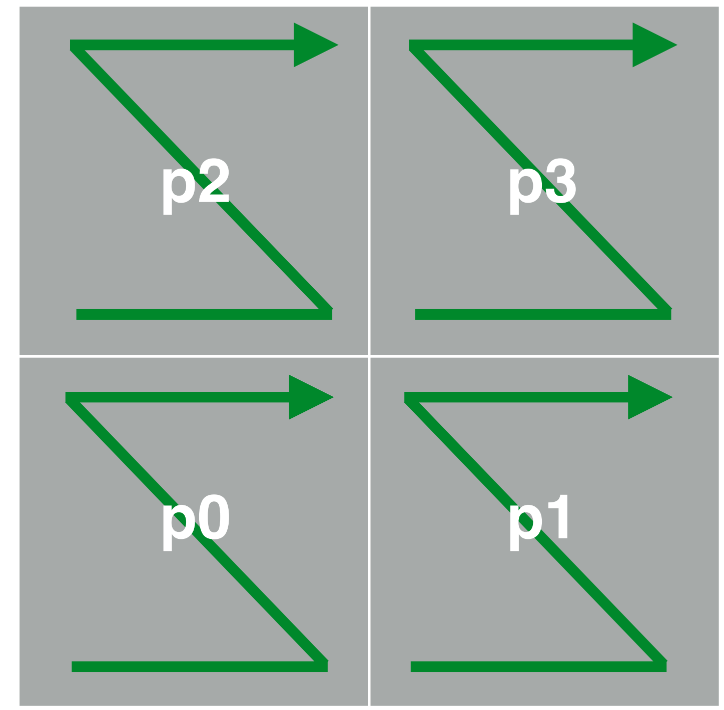
P^T



$A^{16\to 4}$

$b^{16\to 4}$

Repartition: 2 x 2 processors



A^4

b^4

x^4

$$\boldsymbol{A}^{16\to 4} = \boldsymbol{P}^T \boldsymbol{A}^{16} \boldsymbol{P}$$

$$\boldsymbol{A}^4 = \mathrm{GATHER}[\boldsymbol{A}^{16\to 4}]$$

$$\boldsymbol{b}^{16\to 4} = \boldsymbol{P}^T \boldsymbol{b}^{16}$$

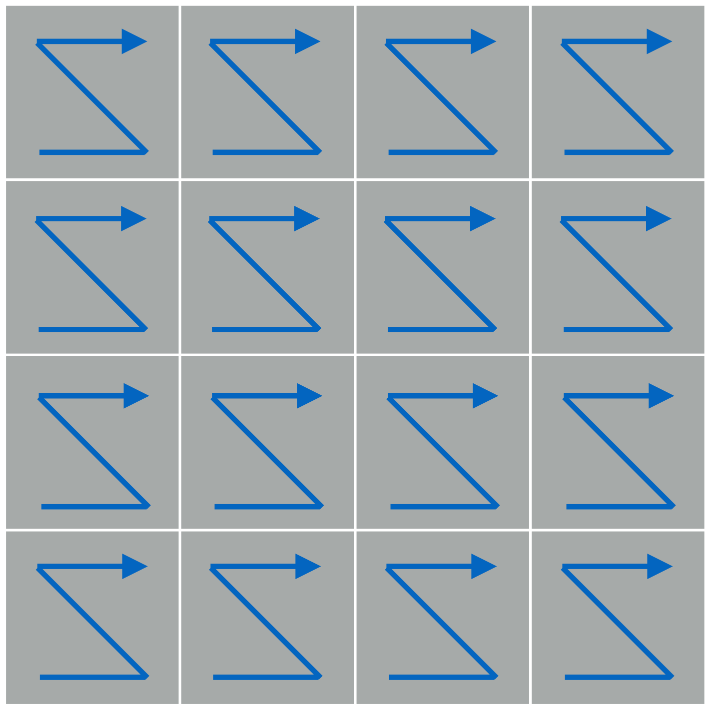
$$\boldsymbol{b}^4 = \mathrm{GATHER}[\boldsymbol{b}^{16\to 4}]$$

Perform solve $\quad \boldsymbol{A}^4 \boldsymbol{x}^4 = \boldsymbol{b}^4$

$$\boldsymbol{x}^{4\to 16} = \mathrm{SCATTER}[\boldsymbol{x}^4]$$

$$\boldsymbol{x}^{16} = \boldsymbol{P}\boldsymbol{x}^{4\to 16}$$

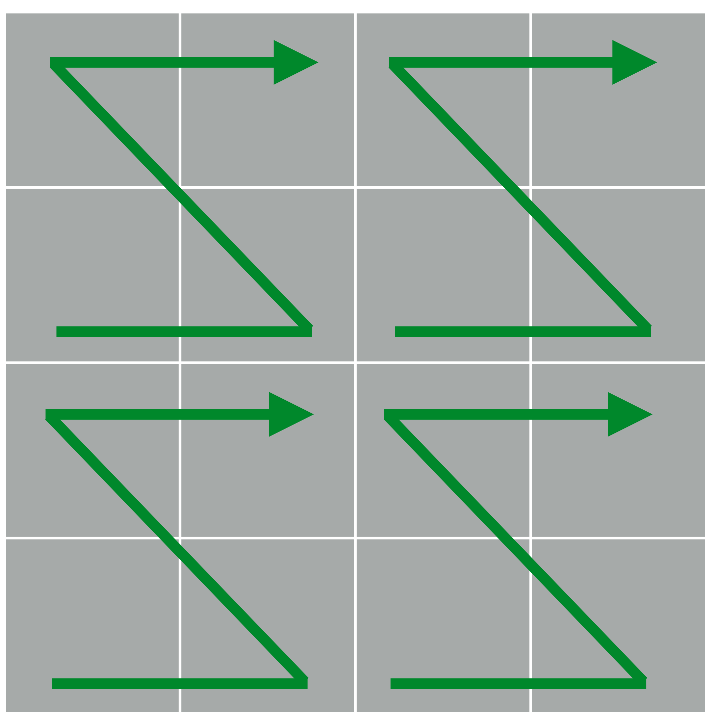
Repartitioning Coarse Grids



A^{16}

b^{16}

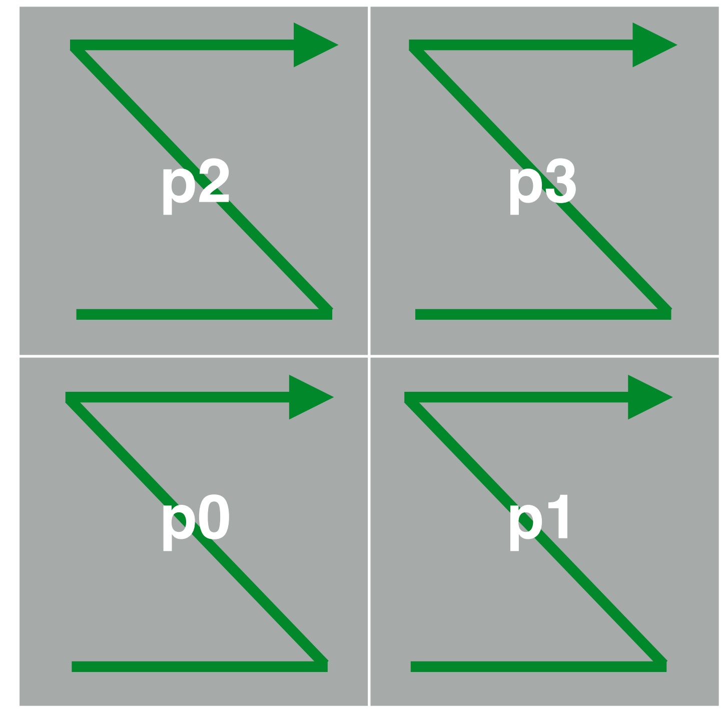
P^T



$A^{16\to 4}$

$b^{16\to 4}$

Repartition: 2 x 2 processors



A^4

b^4

x^4

$$\boldsymbol{A}^{16\to 4} = \boldsymbol{P}^T \boldsymbol{A}^{16} \boldsymbol{P}$$

$$\boldsymbol{A}^4 = \mathrm{GATHER}[\boldsymbol{A}^{16\to 4}]$$

$$\boldsymbol{b}^{16\to 4} = \boldsymbol{P}^T \boldsymbol{b}^{16}$$

$$\boldsymbol{b}^4 = \mathrm{GATHER}[\boldsymbol{b}^{16\to 4}]$$

Perform solve $\quad \boldsymbol{A}^4 \boldsymbol{x}^4 = \boldsymbol{b}^4$

$$\boldsymbol{x}^{4\to 16} = \mathrm{SCATTER}[\boldsymbol{x}^4]$$

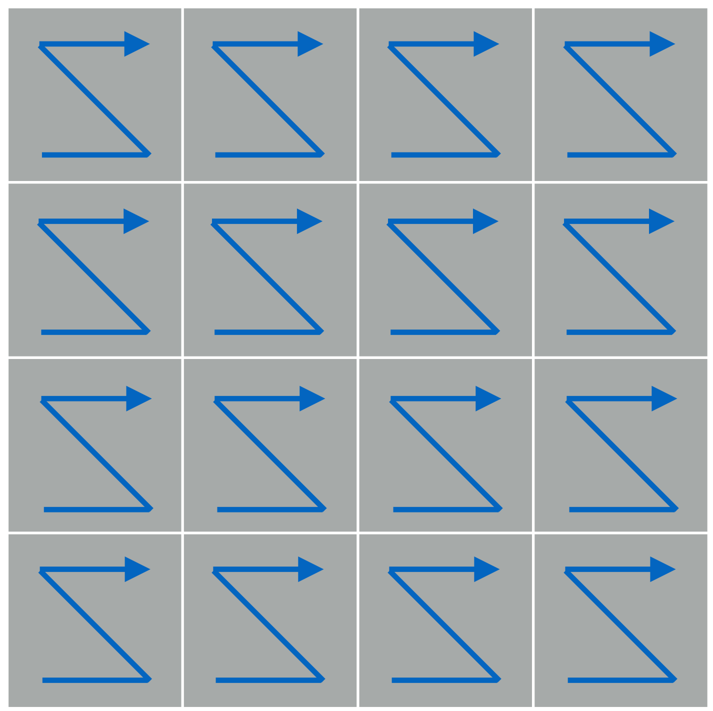
$$\boldsymbol{x}^{16} = \boldsymbol{P} \boldsymbol{x}^{4\to 16}$$

Repartitioning Coarse Grids

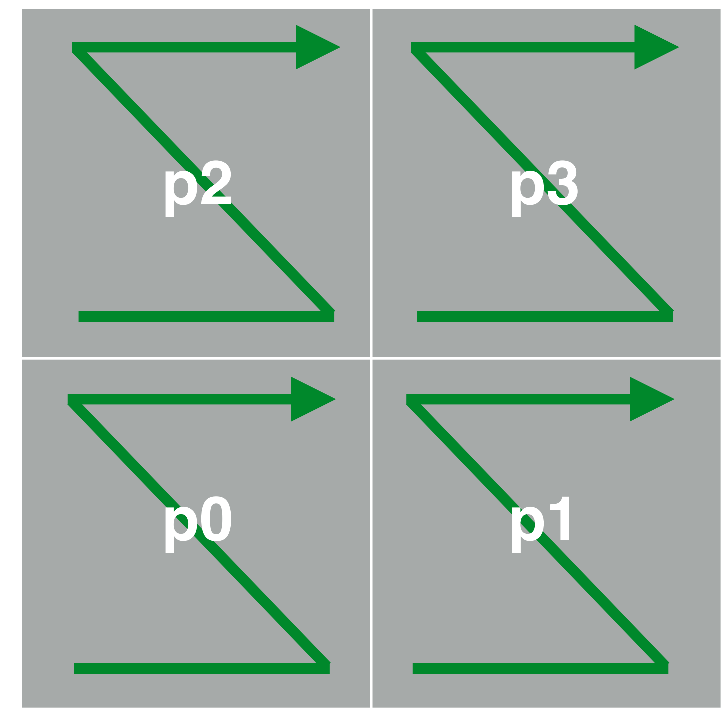
Repartition: 2 x 2 processors

A^{16}

b^{16}



$\xleftarrow{P}$

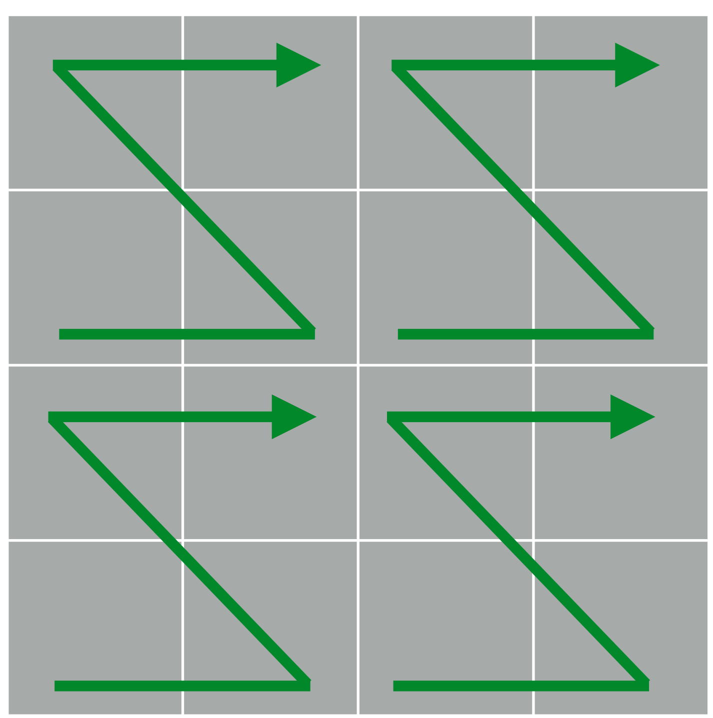


A^4

b^4

x^4

$P^T \downarrow$



$A^{16\to4}$

$b^{16\to4}$

$$\boldsymbol{A}^{16\to4} = \boldsymbol{P}^T\boldsymbol{A}^{16}\boldsymbol{P}$$

$$\boldsymbol{A}^4 = \mathrm{GATHER}[\boldsymbol{A}^{16\to4}]$$

$$\boldsymbol{b}^{16\to4} = \boldsymbol{P}^T\boldsymbol{b}^{16}$$

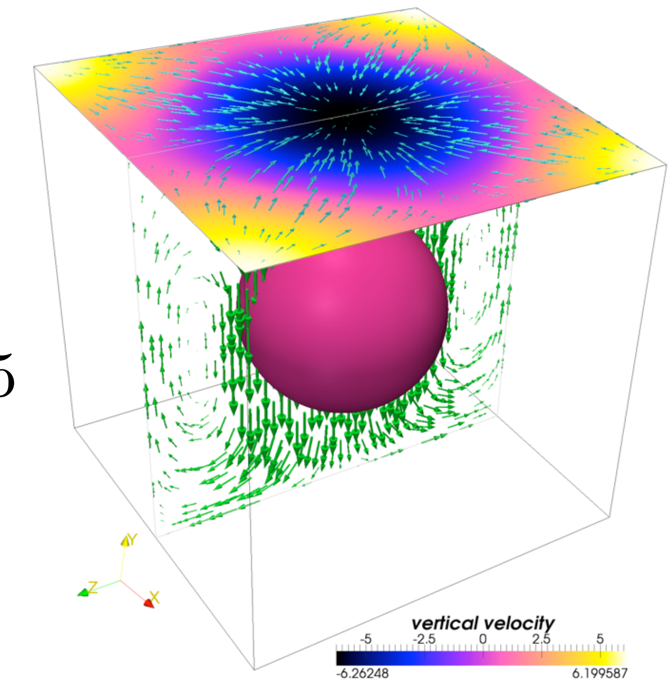
$$\boldsymbol{b}^4 = \mathrm{GATHER}[\boldsymbol{b}^{16\to4}]$$

Perform solve $\quad \boldsymbol{A}^4\boldsymbol{x}^4 = \boldsymbol{b}^4$

$$\boldsymbol{x}^{4\to16} = \mathrm{SCATTER}[\boldsymbol{x}^4]$$

$$\boldsymbol{x}^{16} = \boldsymbol{P}\boldsymbol{x}^{4\to16}$$

Linear Stokes Solver: Strong Scaling



- 96^3 Q2-P1 elements
- 3-level method
- Chebyshev(10)/Jacobi
- Coarse grid solvers:
  - Hierarchical Krylov
  - `PCGAMG`
  - Repartitioned (custom precursor to `PCTelescope`) by a factor of 16

$$R = 0.25$$
$$\Delta\eta = 10^4$$

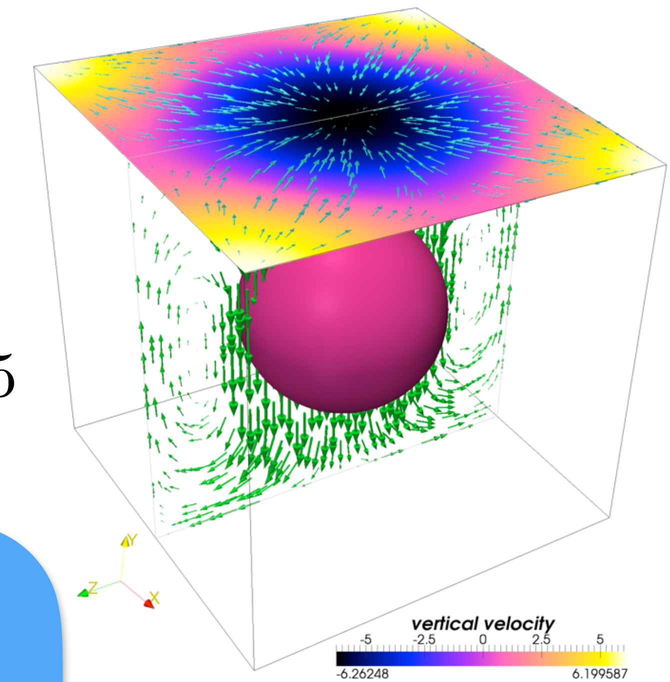
Wall clock times (sec)

MPI-ranks		64	512	4096
Strategy	Task			
H-Krylov	Coarse solve	1.8872e+02	3.3849e+01	9.1787e+00
	Smoother	4.8848e+02	5.1566e+01	7.2146e+00
	Solve	9.9545e+02	1.1651e+02	1.9926e+01
GAMG	Coarse solve	3.3929e+01	4.8522e+00	3.6687e+00
	Smoother	3.4835e+02	4.3411e+01	6.9875e+00
	Solve	5.9950e+02	7.4663e+01	2.1039e+01
Repartitioned	Coarse solve	1.4028e+02	1.7607e+01	2.9893e+00
	Nested coarse solve	1.5587e+01	1.9563e+00	3.3214e−01
	Smoother	3.0379e+02	3.8059e+01	5.6223e+00
	Solve	6.4287e+02	8.0635e+01	1.1826e+01

Linear Stokes Solver: Strong Scaling

- 96^3 Q2-P1 elements
- 3-level method
- Chebyshev(10)/Jacobi
- Coarse grid solvers:
  - Hierarchi[illegible]
  - **PCGAMG**
  - Repartiti[illegible]

$$R = 0.25$$
$$\Delta\eta = 10^4$$




```
-mg_coarse_ksp_type fgmres
-mg_coarse_pc_type ksp
-mg_coarse_ksp_ksp_type chebyshev
-mg_coarse_ksp_ksp_max_it <maxit>
-mg_coarse_ksp_ksp_norm_type none
-mg_coarse_ksp_ksp_convergence_test skip
-mg_coarse_ksp_pc_type <pctype>
```

64 → 4096 ranks

Wall clock times (sec)

MPI-rank[illegible]		[illegible]	[illegible]	12[illegible]
Strategy				
H-Krylov	[illegible]	[illegible]	[illegible]	[illegible]
	Smoother	4.8848e+02	[illegible]	[illegible]
	Solve	9.9545e+02	1.1651e+02	1.9926e+01
GAMG	[illegible]	[illegible]3929e+01	[illegible]	[illegible]
	Smoother	[illegible]4835e+02	[illegible]	[illegible]
	Solve	5.9950e+02	7.4663e+01	2.1039e+01
Repartitio[illegible]	[illegible]	[illegible]e+02	1.7607e+01	2.9893e+00
	[illegible]	[illegible]	[illegible]	[illegible]
	Smoother	3.0379e+02	3.8059e+01	5.6223e+00
	Solve	6.4287e+02	8.0635e+01	1.1826e+01

78% Strong scaling efficiency

```
-mg_coarse_pc_type gamg
```

45% Strong scaling efficiency

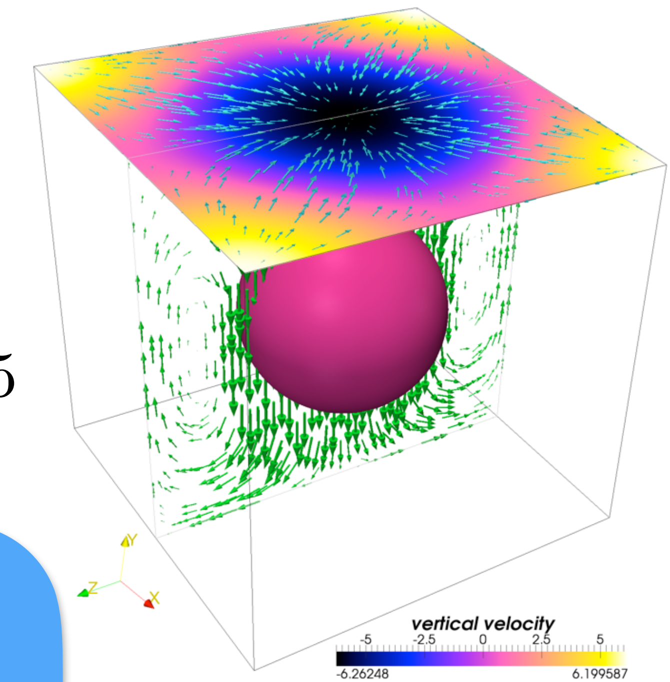
```
-mg_coarse_pc_type ????
```

85% Strong scaling efficiency

Linear Stokes Solver: Strong Scaling

- 96^3 Q2-P1 elements
- 3-level method
- Chebyshev(10)/Jacobi
- Coarse grid solvers:
  - Hierarchi[illegible]
  - **PCGAMG**
  - Repartiti[illegible]

$$R = 0.25$$
$$\Delta\eta = 10^4$$




```
-mg_coarse_ksp_type fgmres
-mg_coarse_pc_type ksp
-mg_coarse_ksp_ksp_type chebyshev
-mg_coarse_ksp_ksp_max_it <maxit>
-mg_coarse_ksp_ksp_norm_type none
-mg_coarse_ksp_ksp_convergence_test skip
-mg_coarse_ksp_pc_type <pctype>
```

64 → 4096 ranks

Wall clock times (sec)

MPI-rank[illegible] Strategy				12[illegible]
H-Krylov	[illegible]	[illegible]	[illegible]	[illegible]
	Smoother	4.8848e[illegible]	[illegible]	[illegible]
	Solve	9.9545e+02	1.1651e+02	1.9926e+01
GAMG	[illegible]	[illegible]3929e[illegible]	[illegible]	[illegible]
	[illegible]	[illegible]4835e[illegible]	[illegible]	[illegible]
	Solve	5.9950e+02	7.4663e+01	2.1039e+01
Repartitio[illegible]	[illegible]	[illegible]e+02	1.7607e+01	2.9893e+00
	[illegible]	[illegible]	[illegible]	[illegible]
	Smoother	3.0379e[illegible]	[illegible]	[illegible]
	Solve	6.4287e+02	8.0635e+01	1.1826e+01

78% Strong scaling efficiency

```
-mg_coarse_pc_type gamg
```

45% Strong scaling efficiency

```
-mg_coarse_pc_type telescope
```

85% Strong scaling efficiency

PCTelescope: *Agglomeration in PETSc*

Flavours of Multigrid for Variable Coefficients

[Chan & Wan, JCP, 2000]

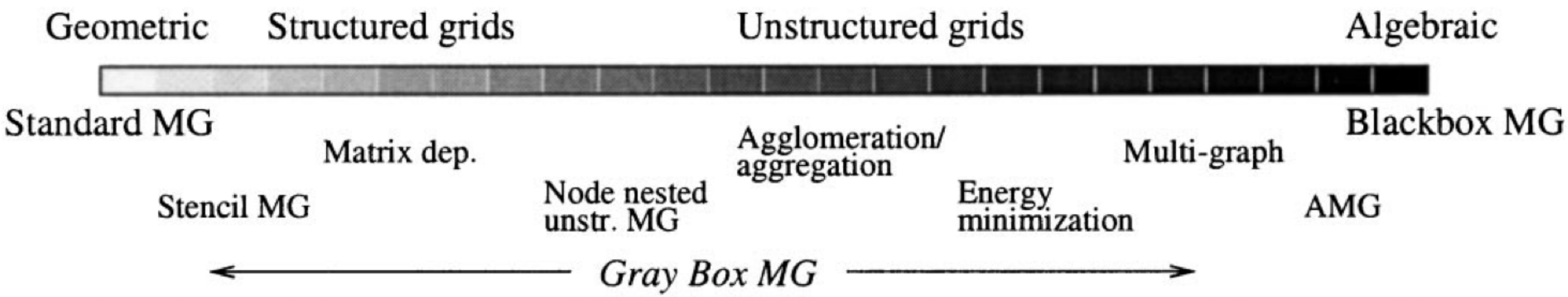


Fig. 1. A spectrum of multigrid methods.

Cheap (“Weak”)

Expensive (“Robust”)

Flavours of Multigrid for Variable Coefficients

[Chan & Wan, JCP, 2000]

Make **modular** and **simple**

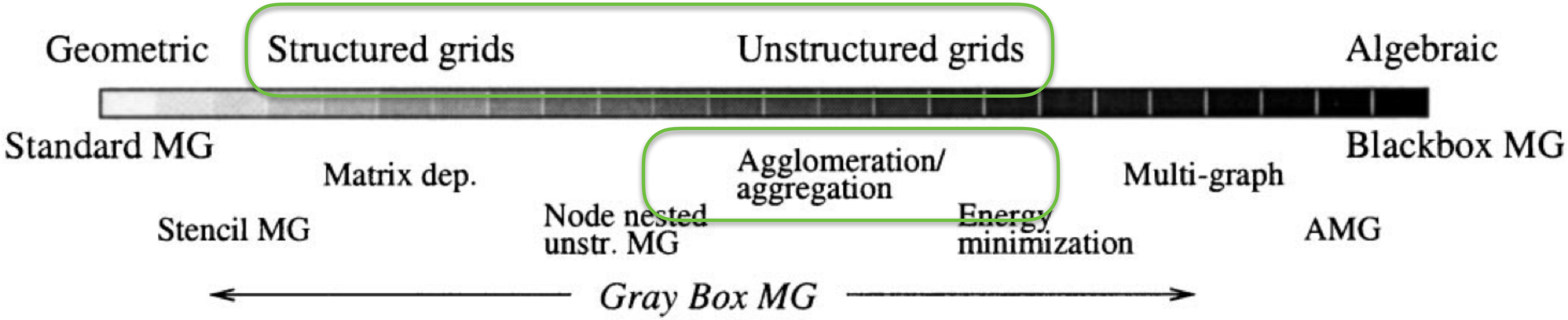


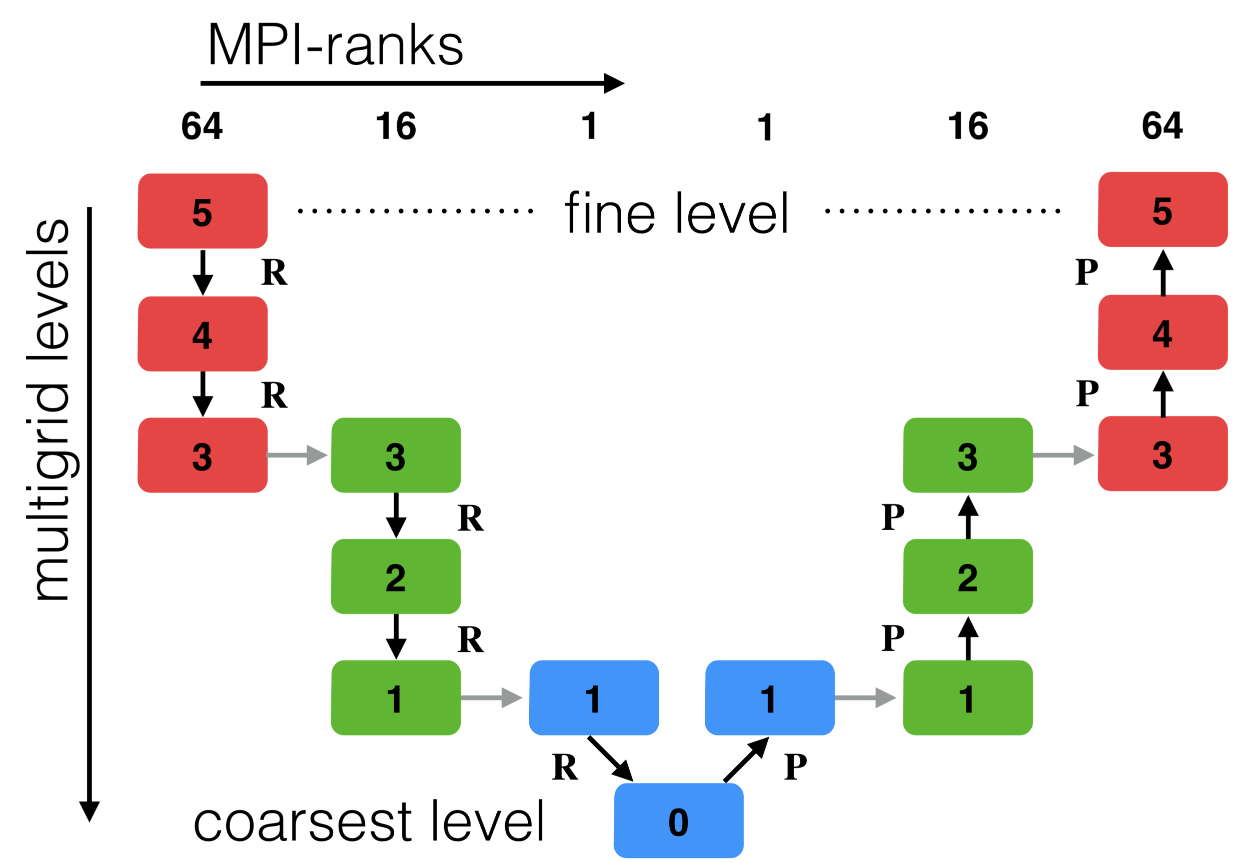
Fig. 1. A spectrum of multigrid methods.

Cheap (“Weak”)

Expensive (“Robust”)

Implementing Agglomeration for Multigrid

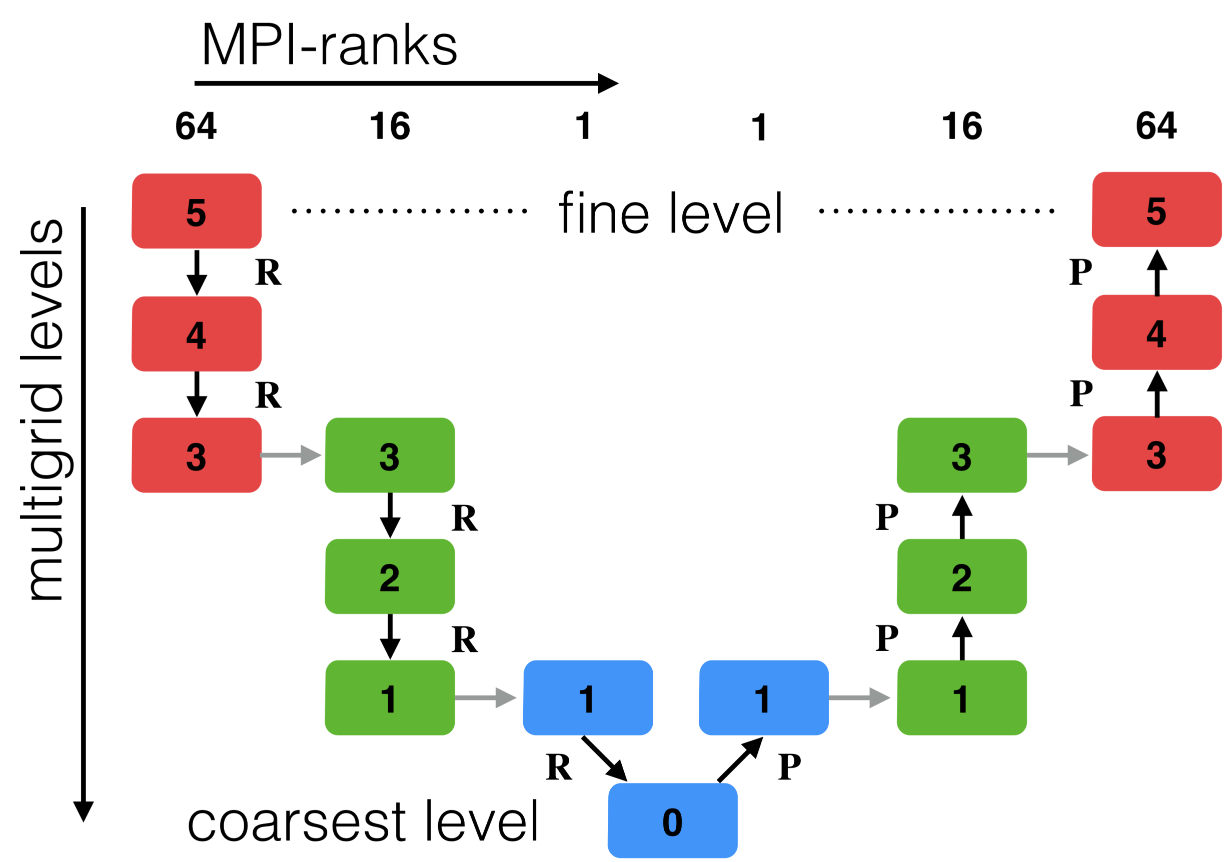
- Not new, not impossible to implement*, but as an extreme-scale component, **rarely implemented at first, and often not at all** if code is insufficiently modular
- Predictive performance models are lacking, so **runtime configurability is useful**
- Agglomeration has uses outside of MG



*See our paper for many references

Implementing Agglomeration for Multigrid

- Not new, not impossible to implement*, but as an extreme-scale component, **rarely implemented at first, and often not at all** if code is insufficiently modular
- Predictive performance models are lacking, so **runtime configurability is useful**
- Agglomeration has uses outside of MG



- We implement agglomeration as a **preconditioner** within PETSc, to provide a reusable building block
  - Simple, composable design
  - Not optimal for all usage, particularly in memory footprint.
- We focus on agglomeration which is **aware of domain connectivity** via PETSc's `DM` class

*See our paper for many references

Design Philosophy

- **P**ortable, **E**xtensible **T**oolkit for **S**cientific **c**omputation
- **P**ortable, **E**xtensible **T**oolkit for **S**olver **c**omposability ?
- Composable building blocks
  - `KSP` : iterative linear solver
  - `PC` : preconditioner within `KSP`
    - Also used for direct solvers
    - Nested `KSP` objects as subsolvers or smoothers
  - `SNES` : nonlinear solver
  - `DM` : domain management
- **Runtime configurability** is a central design decision.
  - experimentation usually required to choose solver parameters
  - Solvers and subsolvers addressed with **options prefixes**

```
-stokes_fieldsplit_u_mg_levels_2_ksp_type sor
```

Anatomy of a Prefix

Name of a **PC** type (**PCSOR**)

Custom prefix for a linear solver (**KSP**)

Option for **PC**

`-stokes_fieldsplit_u_mg_levels_2_pc_type sor`

Prefix for smoother within **PCMG**

Prefix for a block sub-solver within **PCFIELDSPLIT**

DM

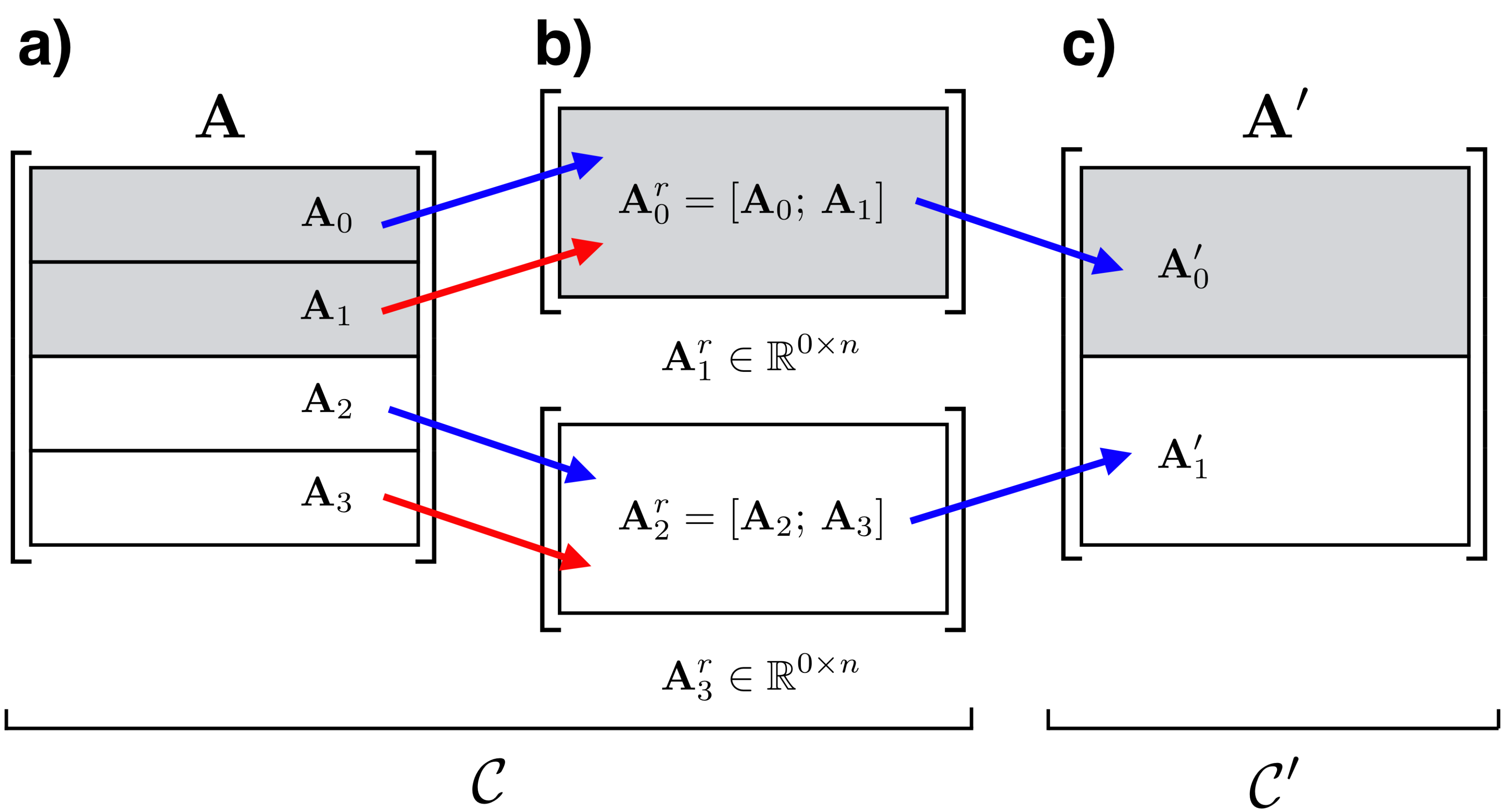
- A class to provide the required interface between solvers and distributed domains
- Geometric primitives, topological relationships between them, and field information

PCMG

- It's not entirely obvious that a solver library should include domain information
- However, geometric multigrid is facilitated with this information, so `PCMG` couples strongly to `DM`
- `PCTelescope` is also "`DM` aware"
- Following the design pattern of providing composable, nestable solvers, the smoothers on each level of the multigrid hierarchy, as well as the coarse grid solver, are `KSP` objects

PCTelescope Design - Assembled Matrices

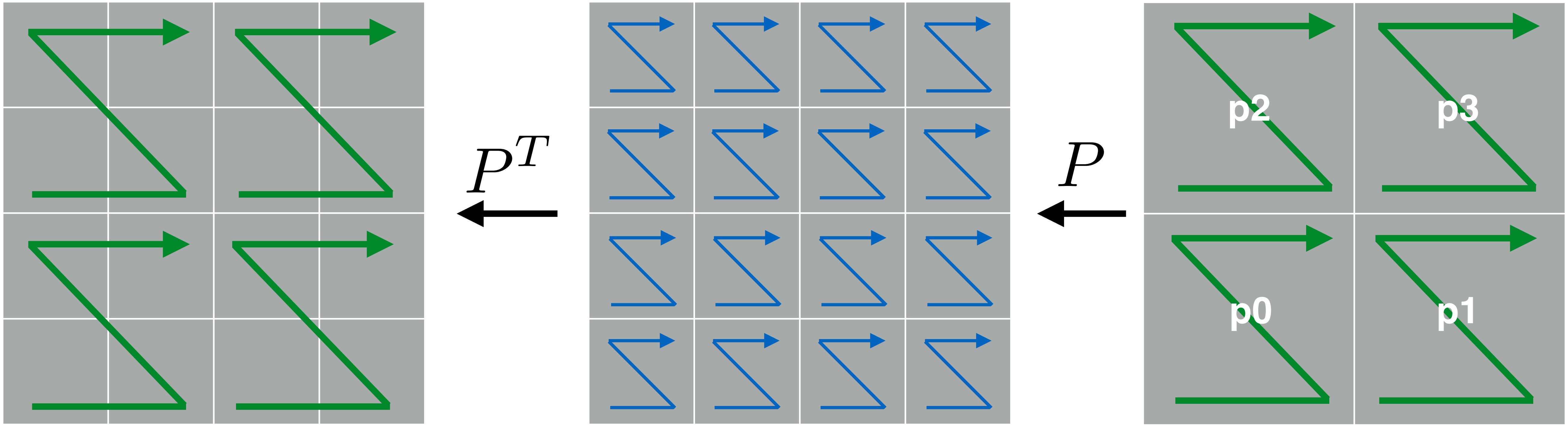
1. Given an MPI communicator $\mathcal{C}$, create a new communicator $\mathcal{C}'$.
2. Repartition the input matrix $\mathbf{A}$ and vector $\mathbf{x}$ onto $\mathcal{C}'$, yielding $\mathbf{A}'$ and $\mathbf{x}'$.
3. Apply a Krylov method to solve $\mathbf{A}'\mathbf{y}' = \mathbf{x}'$ on $\mathcal{C}'$.
4. Scatter the solution $\mathbf{y}'$ to $\mathcal{C}$ to obtain $\mathbf{y}$.



Nullspaces attached to $\mathbf{A}$ are automatically propagated!

DM Repartitioning

- PETSc allows `DM`'s to be associated with `KSP` objects, which in turn makes them available to `PC`'s like `PCTelescope`
- `PCTelescope` can automatically repartition regular 2D and 3D grids represented with `DMDA` objects
- This involves constructing a permutation to account for the new ordering



Use Cases

Multigrid with Truncation

Use an LU routine as a coarse grid solver:

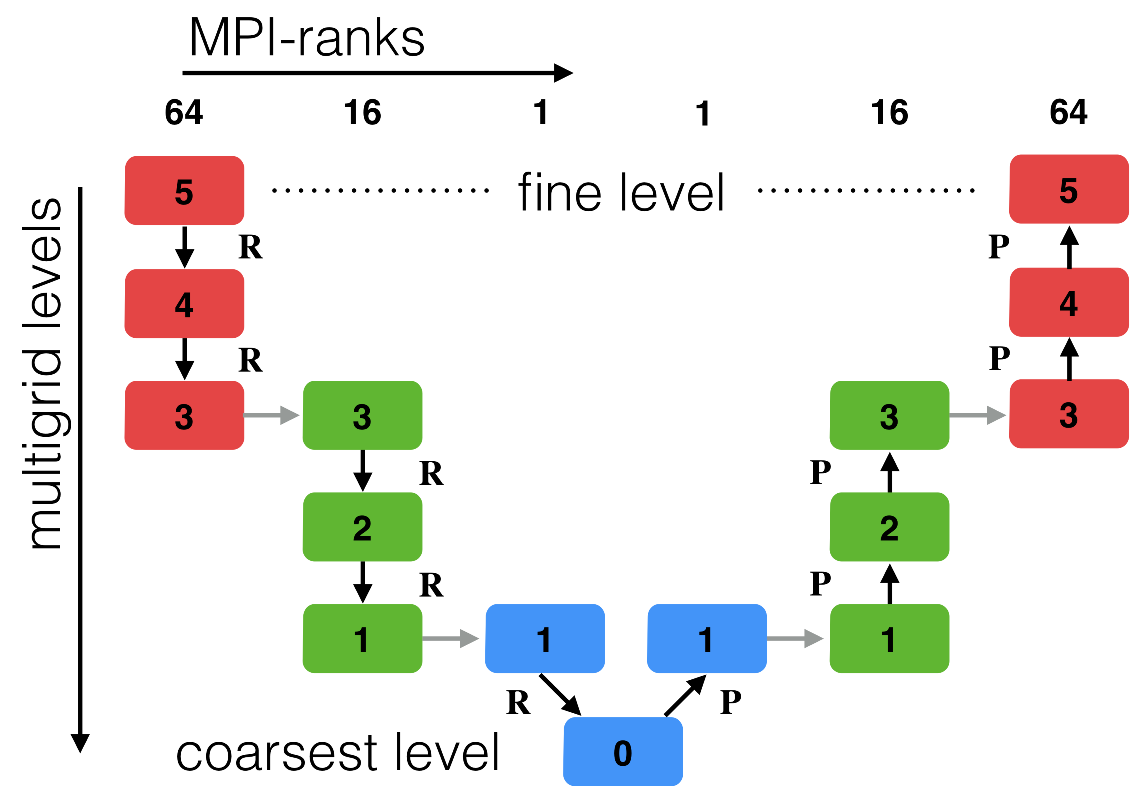
```
-pc_type mg
-pc_mg_levels <N>
-mg_coarse_pc_type telescope
-mg_coarse_pc_telescope_reduction_factor <r>
-mg_coarse_telescope_pc_type lu
-mg_coarse_pc_telescope_subcomm_type
[contiguous,interlaced]
```

Interface to your sequential or parallel direct solver of choice

(recent addition in PETSc `master`)

First **np/r** ranks, or every **r**th rank?

Repartitioned Coarse Grids




```
-pc_type mg
-pc_mg_levels 2
-pc_mg_galerkin
-mg_coarse_pc_type telescope
-mg_coarse_pc_telescope_reduction_factor 4

-mg_coarse_telescope_pc_type mg
-mg_coarse_telescope_pc_mg_levels 2
-mg_coarse_telescope_pc_mg_galerkin
-mg_coarse_telescope_mg_coarse_pc_type telescope
-mg_coarse_telescope_mg_coarse_pc_telescope_reduction_factor 16

-mg_coarse_telescope_mg_coarse_telescope_pc_type mg
-mg_coarse_telescope_mg_coarse_telescope_pc_mg_levels 2
-mg_coarse_telescope_mg_coarse_telescope_pc_mg_galerkin
```

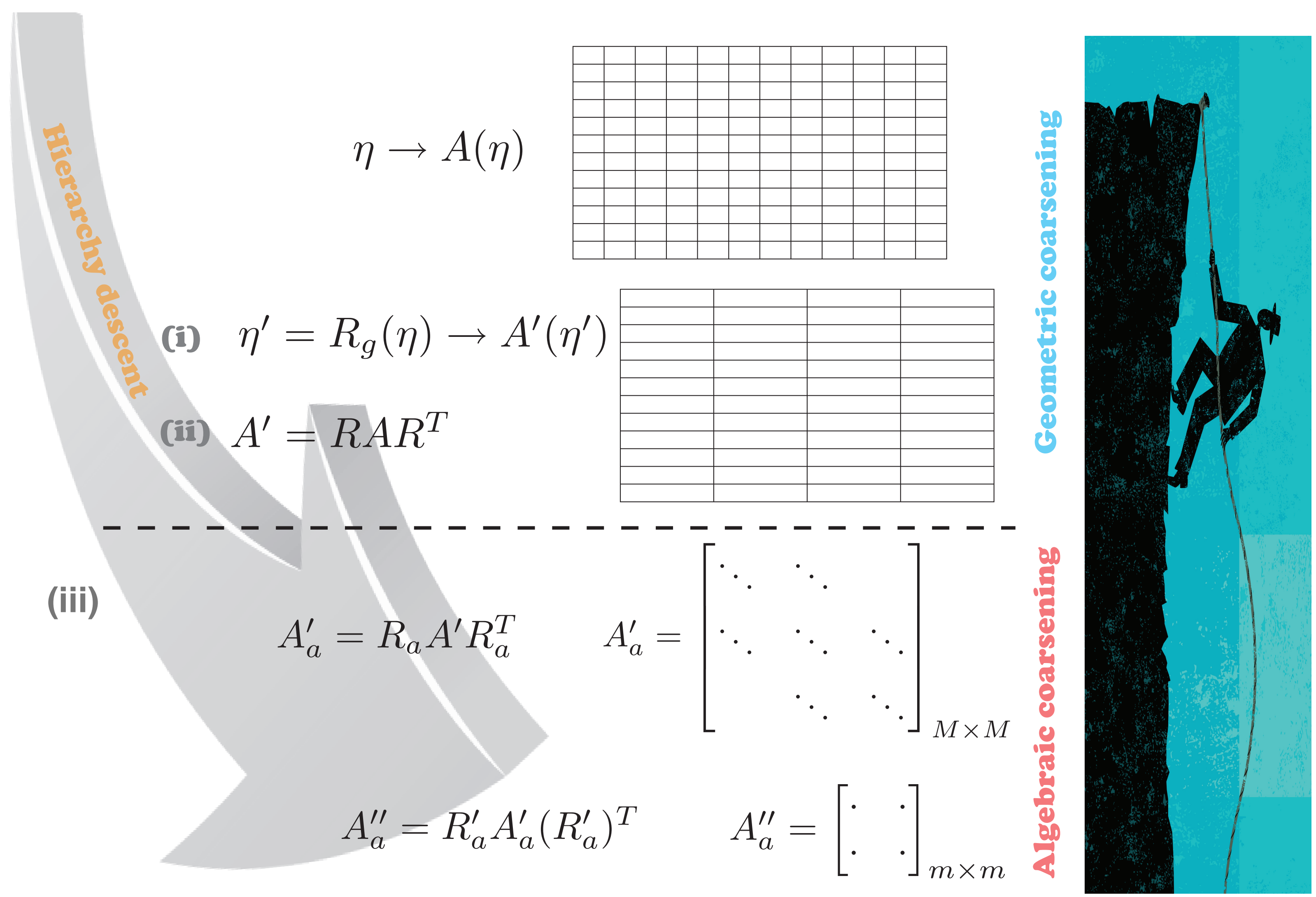
Hybrid Coarse Operator Construction

```
-pc_type mg
-pc_mg_levels <N1>
-mg_coarse_pc_type telescope
-mg_coarse_pc_telescope_reduction_factor <r>
-mg_coarse_telescope_pc_type mg
-mg_coarse_telescope_pc_mg_levels <N2>
-mg_coarse_telescope_pc_mg_galerkin
-mg_coarse_telescope_mg_coarse_pc_type gamg
```

Re-disc. geom. MG

Galerkin MG

Algebraic MG



Subdomain Smoothers with Constant Size

```
-pc_type mg
-pc_mg_levels <N>
-mg_levels_pc_type telescope
-mg_levels_pc_telescope_reduction_factor <rn>
-mg_levels_telescope_pc_type bjacobi
-mg_levels_telescope_sub_pc_type <xxx>
```

Smoothers with Different Spatial Decomposition

```
-pc_type mg
-pc_mg_levels <N>
-mg_levels_pc_type telescope
-mg_levels_pc_telescope_reduction_factor <r>
-mg_levels_telescope_repart_da_processors_z 1
```

Edison

2.57 PFlop/s peak

Numerical Experiments

Piz Daint

7.787 PFlop/s peak

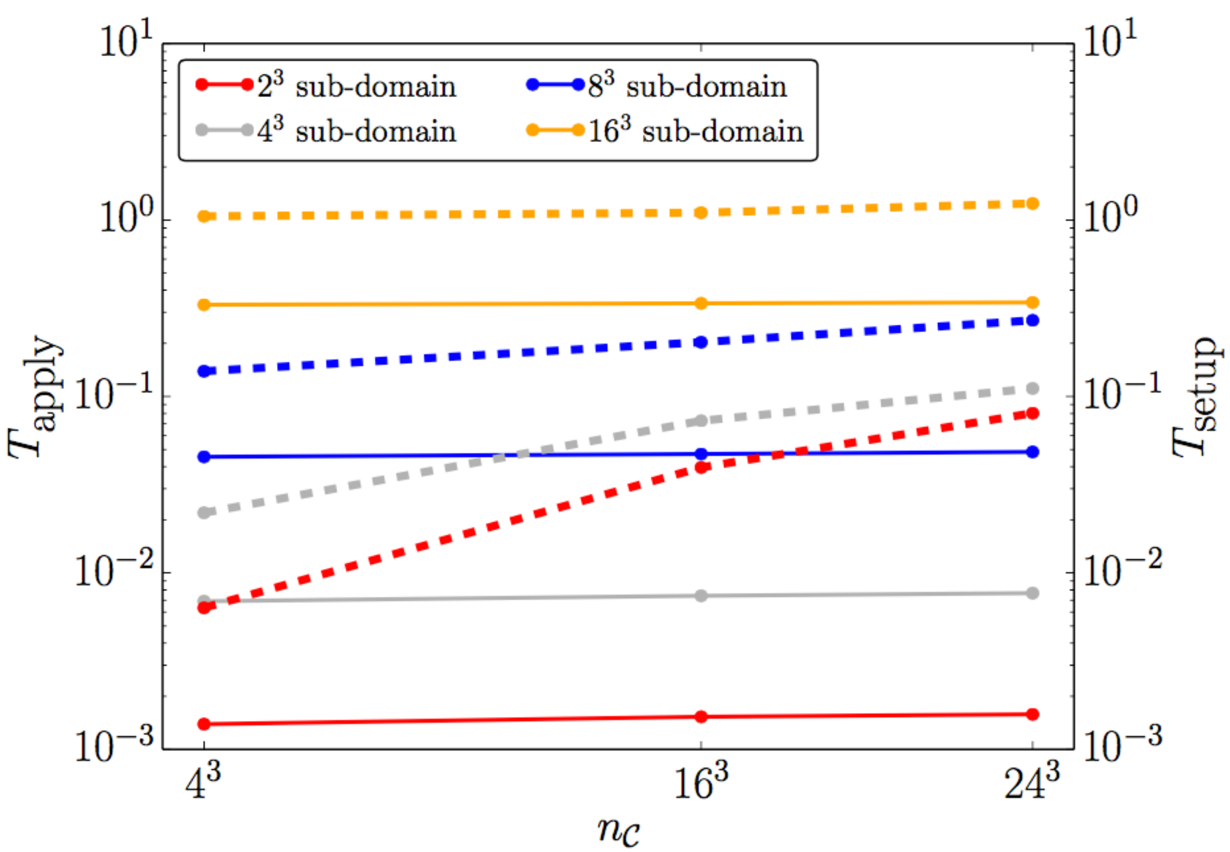
2,968m peak courtesy Sascha M. Schnepp

Agglomeration Profiling

- Profile Setup and Application times for `PCTelescope` on Piz Daint
- 3D FD Laplacian (N^3 DOF)
  - `$PETSC_DIR/src/ksp/ksp/examples/tutorials/ex45.c`

n_C	N	r	T_{setup} (s)	T_{apply} (s)
64	8	8	1.64E−03	8.11E−05
64	8	16	1.77E−03	1.00E−04
64	8	32	1.88E−03	1.51E−04
64	8	64	2.05E−03	2.80E−04
4096	32	8	3.02E−02	5.63E−04
4096	32	16	3.82E−02	3.84E−04
4096	32	32	3.19E−02	3.74E−04
4096	32	64	3.12E−02	6.21E−04
13824	48	8	4.37E−02	4.30E−04
13824	48	16	4.55E−02	3.53E−04
13824	48	32	5.76E−02	5.58E−04
13824	48	64	5.50E−02	5.62E−04

- 3D Q1-Q1 stabilized Stokes problem (M^3 elements)
  - `$PETSC_DIR/src/ksp/ksp/examples/tutorials/ex42.c`



n_C	M	r	T_{setup} (s)	T_{apply} (s)
64	8	8	6.34E−03	1.39E−03
64	8	16	1.02E−02	2.06E−03
64	8	32	1.23E−02	3.26E−03
64	8	64	1.72E−02	4.44E−03
4096	32	8	3.96E−02	1.53E−03
4096	32	16	4.93E−02	2.58E−03
4096	32	32	5.76E−02	4.20E−03
4096	32	64	7.39E−02	7.33E−03
13824	48	8	8.04E−02	1.58E−03
13824	48	16	8.91E−02	2.60E−03
13824	48	32	1.02E−01	4.20E−03
13824	48	64	1.30E−01	7.37E−03

Repartitioning at Scale

- 3D linear elasticity example, run on Edison
- Q2 finite elements implemented on top of `DMDA`
- FGMRES preconditioned with a single V-cycle of geometric multigrid
- Strong-scaling test to stress communication
- “Easy” with constant coefficents: variable coefficients cause further problems for the truncated approach

M	levels	N_L	ranks	$T_{\mathrm{setup}}^{tele}$ (s)	T_{solve} (s)
32	2	2	16^3	–	8.34E−01
32	2, 3	4	16^3, 4^3	8.56E−02	5.23E−01
32	2, 3, 3	6	16^3, 4^3, 1	9.54E−02	1.27E−01
64	2	2	32^3	–	1.48E+01
64	2, 3	4	32^3, 8^3	2.30E−01	1.40E−01
64	2, 3, 3	6	32^3, 8^3, 2^3	3.71E−01	1.82E−01
64	2, 2, 3	5	32^3, 16^3, 4^3	3.43E−01	1.39E−01
64	2, 2, 3, 3	7	32^3, 16^3, 4^3, 1	3.71E−01	1.51E−01

Hybrid CPU-GPU Subdomain Smoothers

- On a hybrid system, one may wish to use agglomerated communicators with a single rank per available accelerator
- We can do so on Piz Daint, assigning a single rank per GPU in the agglomerated communicator
- This allows comparison of SpMV performanceFrom the command line
  - With no need for threads (flat MPI + subcommunicators)

	CPU (8 MPI-ranks)			GPU		
M	Time (s)	GF/s	E/s	Time (s)	GF/s	E/s
4	8.89E−03	11.99	720k	2.43E−02	4.40	264k
8	1.27E−01	6.96	402k	5.90E−02	14.99	865k
12	4.15E−01	7.3	417k	1.91E−01	15.91	908k
24	3.15E+00	7.79	439k	1.44E+00	17.09	963k

Hybrid CPU-GPU Subdomain Smoothers

- We can also compare time to solution of a full solve using GPU subdomain smoothers

M	levels	overlap	T_{setup} (s)	Its.	T_{solve} (s)
8	2	–	1.12E−02	12	4.27E−02
12	3	–	4.41E−02	16	2.06E−01
24	3	–	1.88E−01	13	1.55E+00
48	4	–	1.29E+00	11	9.92E+00
8	2	0	5.49E−01	12	2.2813e-01
12	2	0	2.52E+00	16	2.3985e-01
24	3	0	4.94E+00	13	**1.28E+00**
48	4	0	3.58E+01	11	**6.66E+00**
8	2	1	5.95E−01	12	2.40E−01
12	2	1	1.10E+00	16	4.30E−01
24	3	1	5.55E+00	13	**1.52E+00**
48	4	1	2.30E+01	11	**7.34E+00**

Future Development: Agglomeration for Multigrid on Unstructured Meshes

Extending to Support Unstructured Grids

- PETSc supports unstructured grids via the `DMPlex` class
- Ordering is more complicated
  - “Reduction factor” is less clear
  - Permutation and Scatter objects more complex to generate
- More attached structure must be considered and repartitioned
- Regardless, all required operations are algebraic and can be defined - the key is to lower the burden on a typical user
- Proposed Solution
  - When working with `DMPlex` (or more exotic `DM` implementations), return the responsibility of defining the reduced communicator and required mappings to the `DM`, requiring a call to `DMPlexGetReducedComm()`

Concluding Remarks

- Subdomain agglomeration in extreme-scale geometric multigrid allows for scalability
- This pattern can be encapsulated as a component with preconditioner semantics
- A single simple design, aware of operator nullspaces and underlying domain descriptions, can be effectively used in several ways
  - Coarse grid agglomeration in multigrid
  - Efficient construction of agglomerated subdomains to use with factorization-based sub-solvers
  - Efficient construction of agglomerated subdomains for use with coprocessors associated with multiple CPU cores in a flat MPI environment

Concluding Remarks

`PCTelescope` available in PETSc 3.7

- Composable tool for MPI rank agglomeration, implemented as a PETSc `PC`
- Aware of operator nullspaces and structured grids (`DMDA`)
- Useful for multigrid hierarchies as well as other tasks requiring agglomeration
- Controllable at runtime from the command line
- Main use case: (hybrid) MG hierarchies
- Auxiliary use cases: easy plumbing to define nested operators
- Also supports matrix-free / unassembled operators
  - Override `DMCreateMatrix()` and use `KSPSetComputeOperators()`

Thank You for Your Attention, and Try It Out!

- `PCTelescope` in current PETSc release 3.7.x
  - `mcs.anl.gov/petsc`
- Ongoing improvements in PETSc **`master`**
  - `https://bitbucket.org/petsc/petsc`
- Get in touch if you are interested in the development of `PCTelescope` for unstructured meshes used `DMPlex`
  - `dave.may@erdw.eth.ch`
  - `patrick.sanan@{usi.ch,erdw.ethz.ch}`
- Paper:
  - Dave A. May, Patrick Sanan, Karl Rupp, Matthew G. Knepley, and Barry F. Smith. 2016. **Extreme-Scale Multigrid Components within PETSc**. In Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '16)

ETH zürich

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Università della Svizzera italiana
Faculty of Informatics
Institute of Computational Science ICS

Platform for Advanced Scientific Computing

GEOPC