Argonne Training Program on Extreme-Scale Computing

ATPESC 2020

Krylov Solvers and Algebraic Multigrid with *hypre*

Ulrike Meier Yang
Lawrence Livermore National Laboratory

July 26 – August 7, 2020

Outline

- What are Krylov Solvers?
- Why are they used?
- Why multigrid methods?
- Algebraic multigrid software
- Hypre software library – interfaces
  - Why different interfaces?
- How does multigrid work?
- Unstructured and structured multigrid solvers

Iterative Solvers

- Solve linear system $Ax = b$,
  where A is a large sparse matrix of size n
- Direct solvers (e.g. Gaussian elimination) too expensive
- Iterative solvers
- Richardson iteration:

$$x^{n+1} = x^n + (b - Ax^n)$$
$$e^{n+1} = (I - A)e^n$$

- Introduce a preconditioner B:

$$x^{n+1} = x^n + B(b - Ax^n)$$
$$e^{n+1} = (I - BA)e^n$$

- Jacobi: $B = D^{-1}$;   Richardson: $B = \lambda I$

Generalized Minimal Residual (GMRES)

- $x^{n+1} = x^n + B(b - Ax^n)$
- $\Rightarrow x^{n+1} = \sum_{i=0}^{n} \alpha_i (BA)^i Bb$
- $x^{n+1} \in K^n = span\{Bb, (BA)Bb, (BA)^2Bb, \ldots, (BA)^n Bb\}$

  Krylov space

- Now optimize by defining x^{n+1} through

$$\min_{x^{n+1} \in K^n} ||B(Ax^{n+1} - b)||$$

- Construct a new basis for K^n through orthonormalization

$$\{q_0 = \frac{Bb}{||Bb||}, q_1, \ldots, q_n\}$$

- Solve the minimization in the new basis
- q_i also called search directions

Some comments on GMRES

- GMRES consists of fairly simple operations:
  - Inner products and norms (global reductions)
  - Vector updates (embarrassingly parallel)
  - Matvecs (nearest neighbor updates)
  - Application of preconditioner (can be very complicated)
- Often used restarted as GMRES(k), i.e. after k iterations throw out q_i and start again using latest approximation
- Many variants to reduce and/or overlap communication (pipelined GMRES, etc)

Other Krylov solvers

- Conjugate Gradient (CG)
  - For symmetric positive definite matrices
  - Possesses like GMRES an orthogonality property
  - Uses a three-term concurrence
  - Requires only two inner products and a norm per iteration
- Biconjugate Gradient Stabilized (BiCGSTAB)
  - Like CG uses a three-term recurrence relation
  - No orthogonality property, can break down
  - Requires several inner products and a norm at each iteration (and two matvecs)
  - More erratic convergence than GMRES, but needs generally less memory

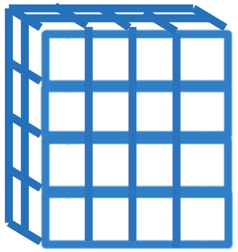
Hands-on Exercises: Krylov methods

- Go to https://xsdk-project.github.io/MathPackagesTraining2020/lessons/krylov_amg_hypre/

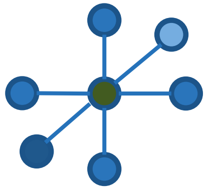
- Poisson equation: $-\Delta\varphi$ = RHS

  with Dirichlet boundary conditions $\varphi = 0$

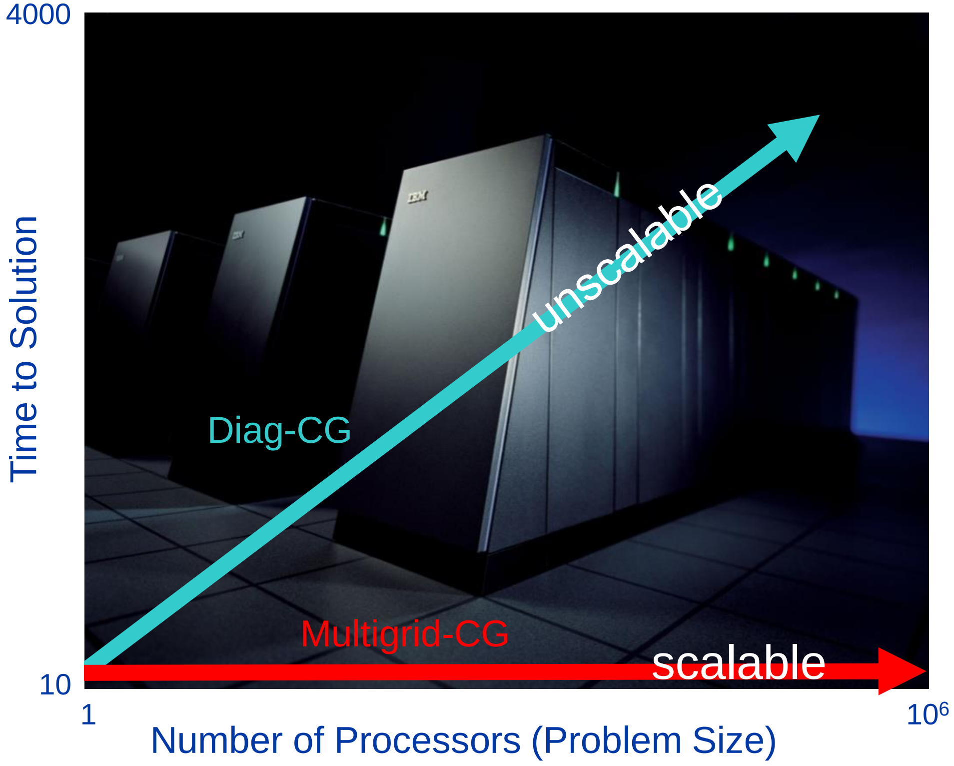
- Grid: cube



- Finite difference discretization:
  - Central differences for diffusion term
  - 7-point stencil



Multigrid linear solvers are optimal ($O(N)$ operations), and hence have good scaling potential

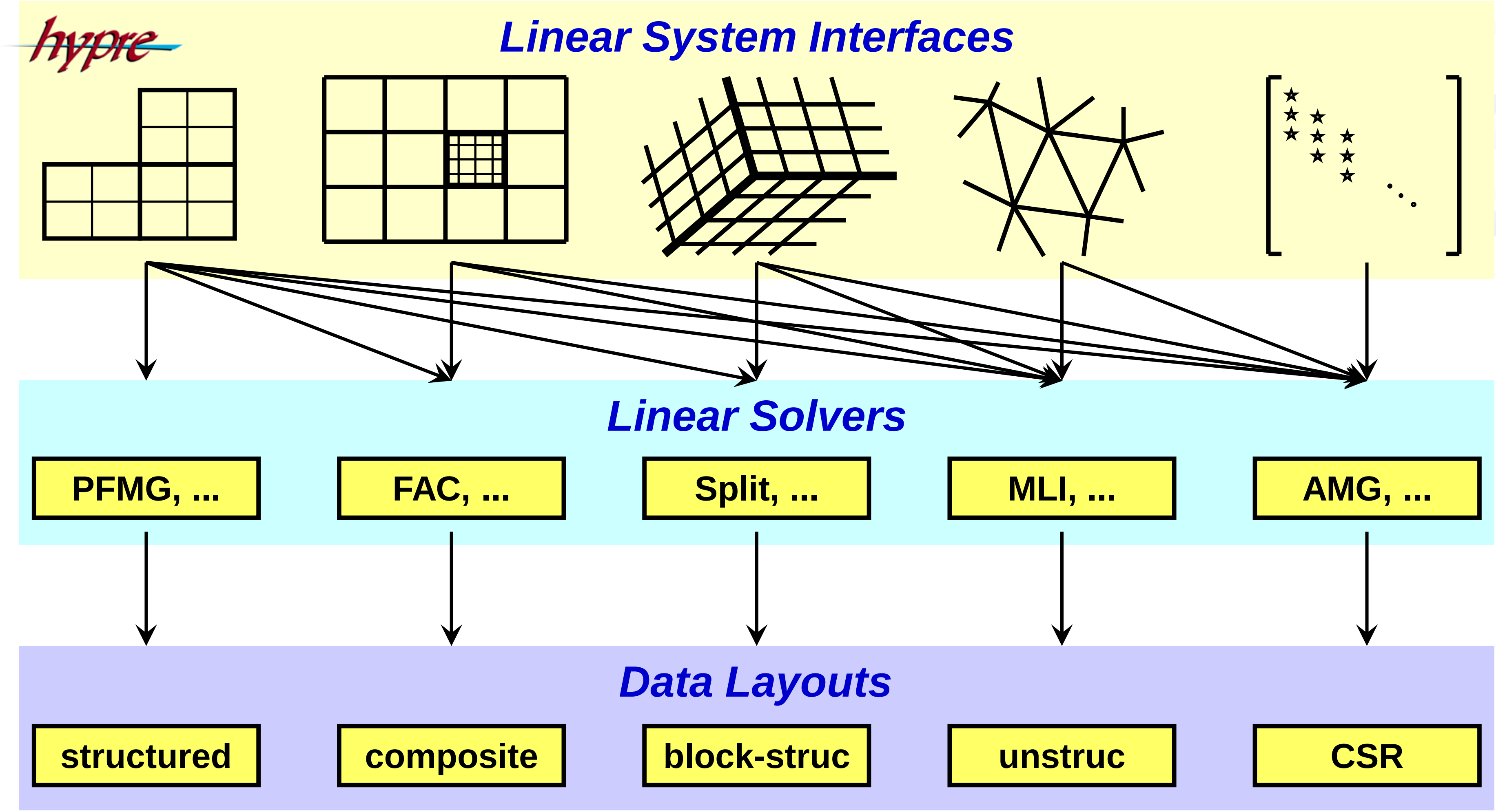


- Weak scaling – want constant solution time as problem size grows in proportion to the number of processors

Available multigrid software

- ML, MueLu included in TRILINOS
- GAMG in PETSc
- The hypre library provides various algebraic multigrid solvers, including multigrid solvers for special problems e.g. Maxwell equations, …
- All of these provide different flavors of multigrid and provide excellent performance for suitable problems
- Focus here on hypre

(Conceptual) linear system interfaces are necessary to provide “best” solvers and data layouts

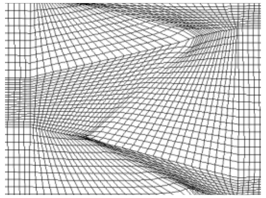


Why multiple interfaces? The key points

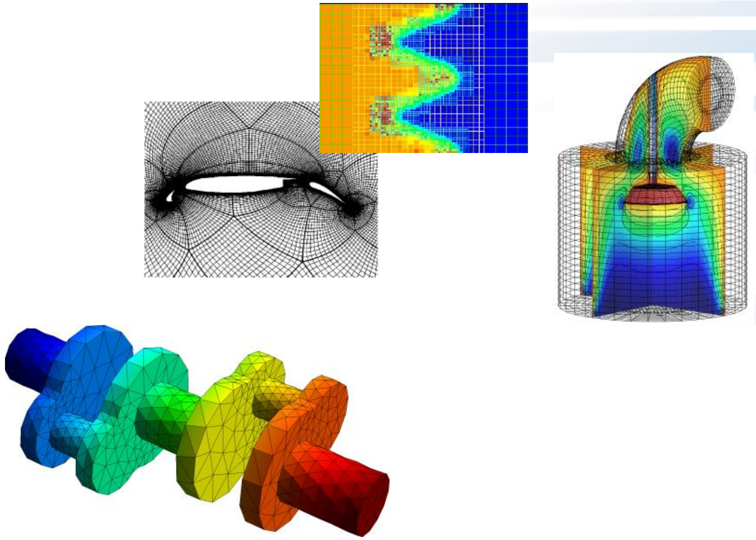
- Provides natural “views” of the linear system
- Eases some of the coding burden for users by eliminating the need to map to rows/columns
- Provides for more efficient (scalable) linear solvers
- Provides for more effective data storage schemes and more efficient computational kernels

hypre supports these system interfaces

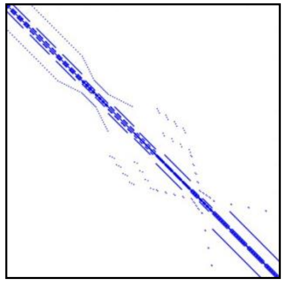
- Structured-Grid (`Struct`)
  - *logically rectangular grids*



- Semi-Structured-Grid (`SStruct`)
  - *grids that are mostly structured*
  - *Examples: block-structured grids, structured adaptive mesh refinement grids, overset grids*
  - *Finite elements*

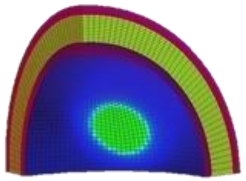


- Linear-Algebraic (`IJ`)
  - *general sparse linear systems*

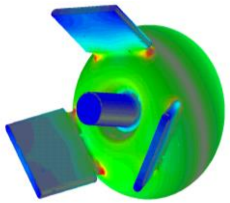


The *hypre* software library provides structured and unstructured multigrid solvers

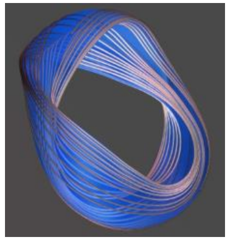
- Used in many applications



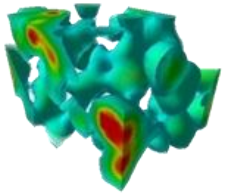
Elasticity / Plasticity



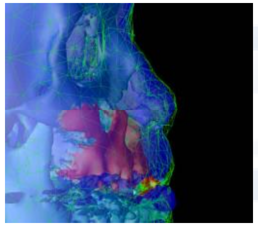
Electro-magnetics



Magneto-hydrodynamics

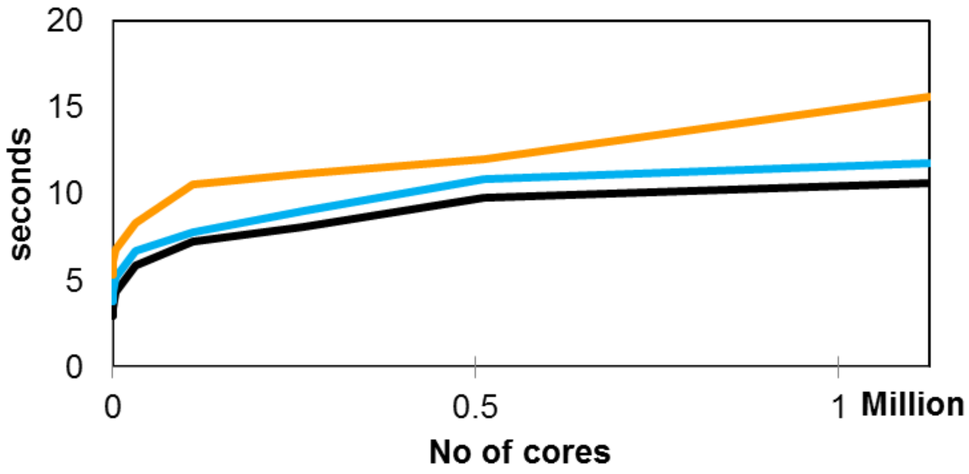


Quantum Chromodynamics

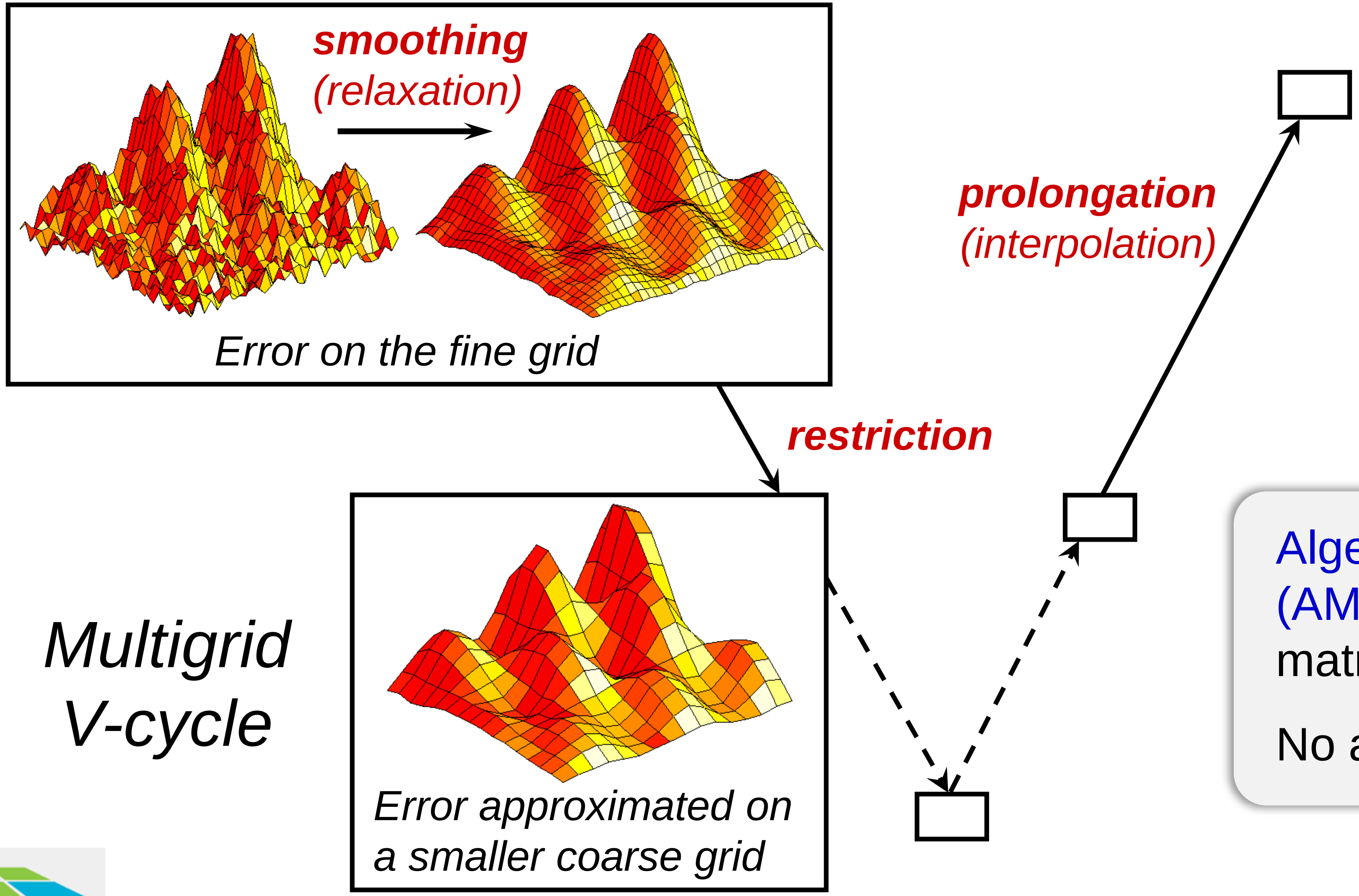


Facial surgery

- Displays excellent weak scaling and parallelization properties on BG/Q type architectures



Multigrid (MG) uses a sequence of coarse grids to accelerate the fine grid solution



Algebraic multigrid (AMG) only uses matrix coefficients

No actual grids!

AMG Building Blocks

Setup Phase:

- **Select coarse "grids"**
- **Define interpolation:** $P^{(m)}, \quad m = 1,2,\ldots$
- **Define restriction:** $R^{(m)}, \quad m = 1,2,\ldots, \quad \text{often } R^{(m)} = (P^{(m)})^T$
- **Define coarse-grid operators:** $A^{(m+1)} = R^{(m)} A^{(m)} P^{(m)}$

  Galerkin product

Solve Phase:

Relax $A^{(m)} u^m = f^m$

Compute $r^m = f^m - A^{(m)} u^m$

Restrict $r^{m+1} = R^{(m)} r^m$

Solve $A^{(m+1)} e^{m+1} = r^{m+1}$

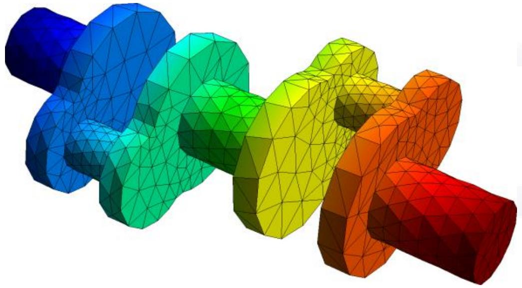
Interpolate $e^m = P^{(m)} e^{m+1}$

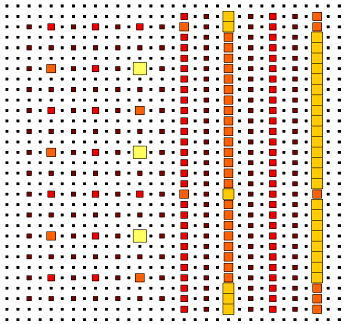
Correct $u^m \leftarrow u^m + e^m$

Relax $A^{(m)} u^m = f^m$

BoomerAMG is an algebraic multigrid method for unstructured grids

- Interface: `SStruct, IJ`
- Matrix Class: `ParCSR`
- Originally developed as a general matrix method (i.e., assumes given only A, x, and b)
- Various coarsening, interpolation and relaxation schemes
- Automatically coarsens “grids”
- Can solve systems of PDEs if additional information is provided
- Can also be used through PETSc and Trilinos
- Can now also be used on GPUs (limited options)



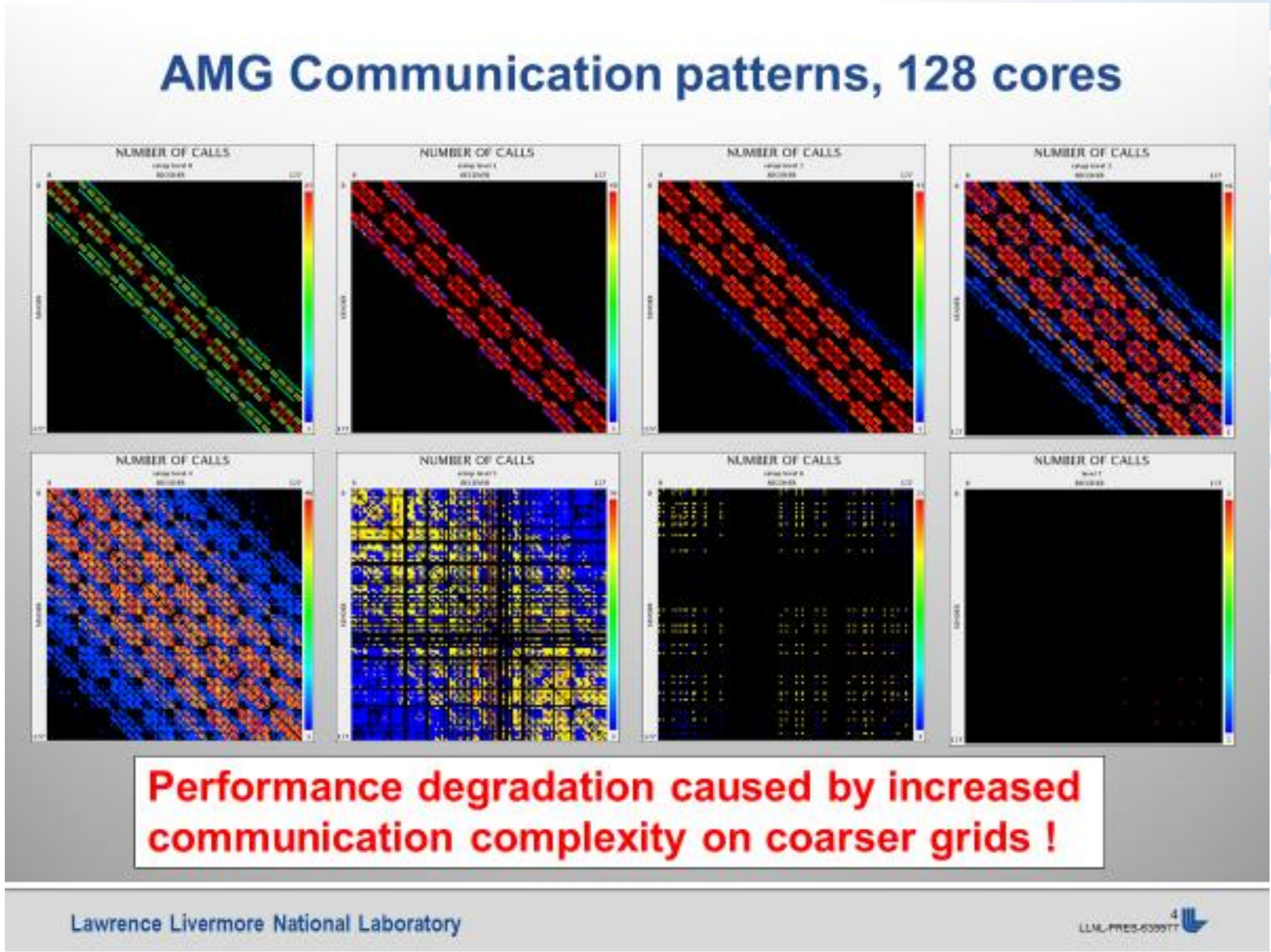


FASTMATH

EXASCALE COMPUTING PROJECT

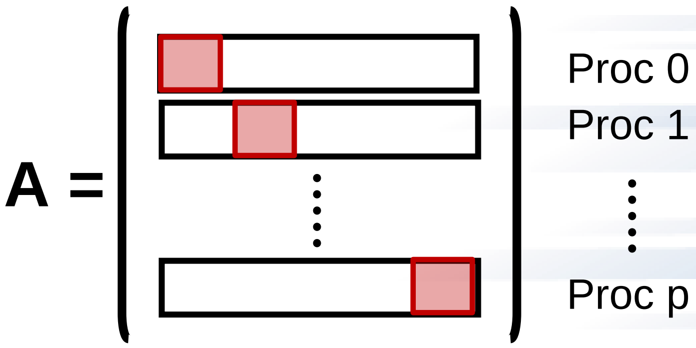
Complexity issues

- Coarse-grid selection in AMG can produce unwanted side effects
- Operator (RAP) “stencil growth” reduces efficiency
- For BoomerAMG we will therefore also consider complexities:
  - Operator complexity: $C_{op} = (\sum_{i=0}^{L} nnz\,(A_i))/nnz(A_0)$
  - Affects flops and memory
  - Generally would like $C_{op} < 2$, close to 1
- Can control complexities in various ways
  - varying strength threshold
  - more aggressive coarsening
  - Operator sparsification (interpolation truncation, non-Galerkin approach)
- Needs to be done carefully to avoid excessive convergence deterioration



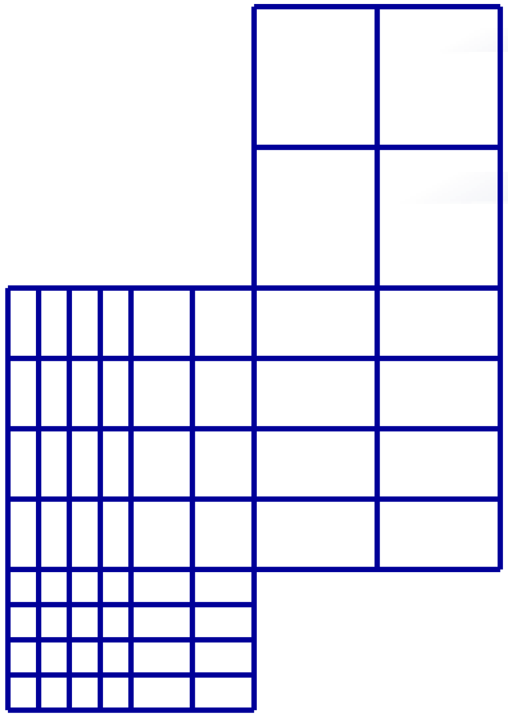
ParCSRMatrix data structure

- Based on compressed sparse row (CSR) data structure
- Consists of two CSR matrices:
  – One containing local coefficients connecting to local column indices
  – The other (Offd) containing coefficients with column indices pointing to off processor rows
- Also contains a mapping between local and global column indices for Offd
- Requires much indirect addressing, integer computations, and computations of relationships between processes etc,

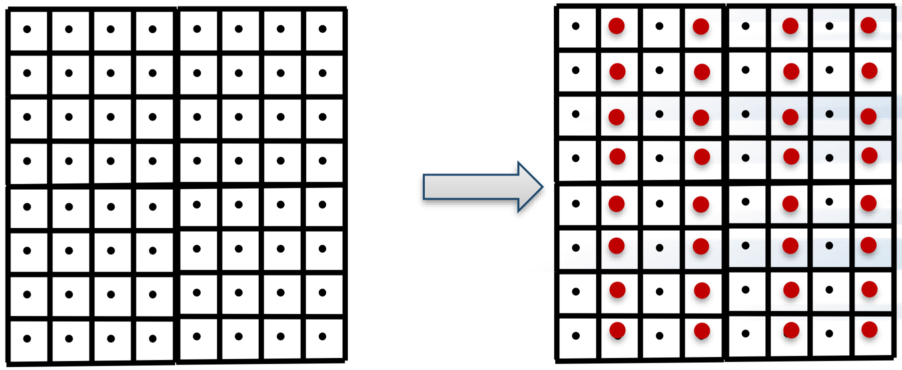


SMG and PFMG are semicoarsening multigrid methods for structured grids

- Interface: `Struct, SStruct`
- Matrix Class: `Struct`
- SMG uses plane smoothing in 3D, where each plane “solve”
  is effected by one 2D V-cycle
- SMG is very robust
- PFMG uses simple pointwise smoothing, and is less robust
- Note that stencil growth is limited for SMG and PFMG (to at most 27 points per stencil in 3D)
- Constant-coefficient versions
- Can be used on GPUs (CUDA, RAJA, Kokkos)



PFMG is an algebraic multigrid method for structured grids

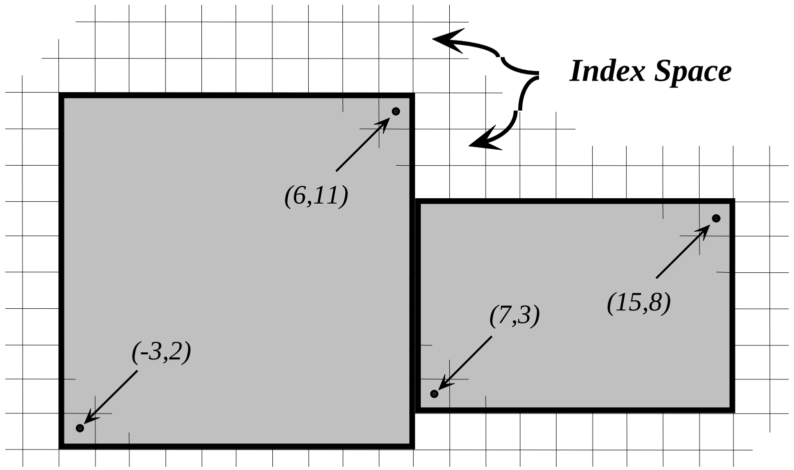


- Matrix defined in terms of grids and stencils
- Uses semicoarsening
- Simple interpolation
  → limits stencil growth to at most 9pt (2D), 27pt (3D)
- Optional non-Galerkin approach (Ashby, Falgout), uses geometric knowledge, preserves stencil size
- Pointwise smoothing
- Highly efficient for suitable problems

Structured-Grid System Interface (`Struct`)

- Appropriate for scalar applications on structured grids with a fixed stencil pattern
- Grids are described via a global *d*-dimensional *index space* (singles in 1D, tuples in 2D, and triples in 3D)
- A *box* is a collection of cell-centered indices, described by its “lower” and “upper” corners
- The grid is a collection of boxes
- Matrix coefficients are defined via stencils

$$\begin{bmatrix} & S4 & \\ S1 & S0 & S2 \\ & S3 & \end{bmatrix} = \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}$$



FASTMATH

EXASCALE COMPUTING PROJECT

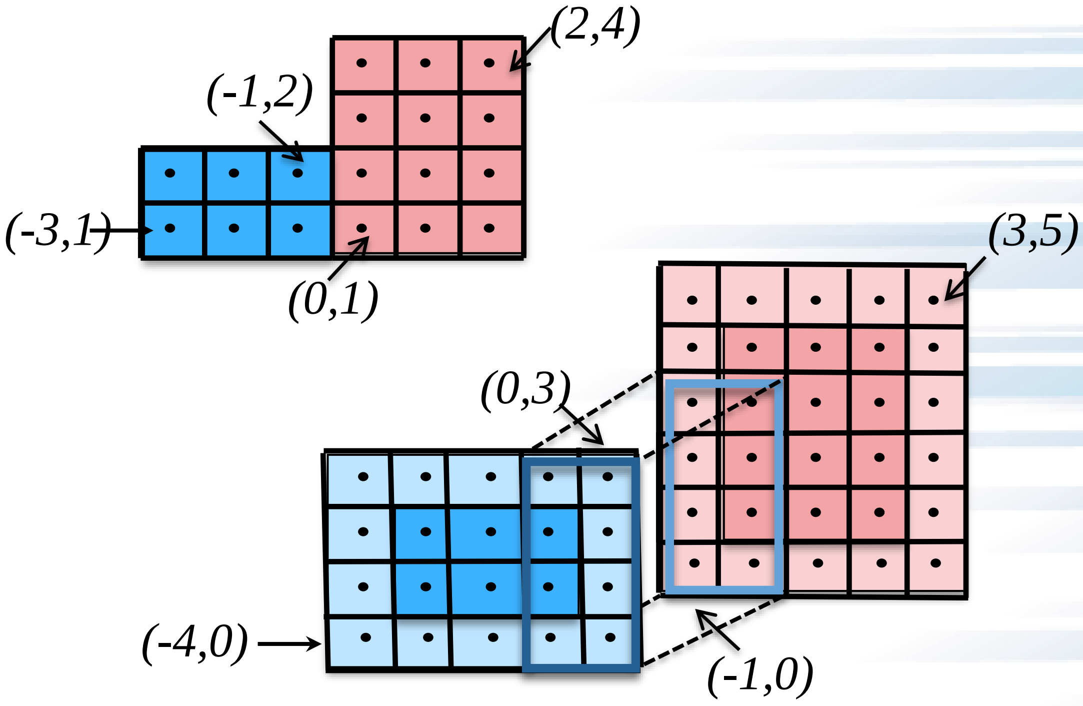
StructMatrix data structure

- Stencil

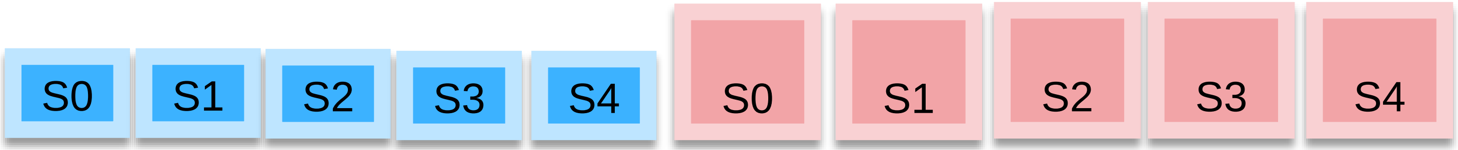
$$\begin{array}{ccc} & S4 & \\ \left[S1 \right. & S0 & \left. S2 \right] \\ & S3 & \end{array} = \begin{array}{ccc} & -1 & \\ \left[-1 \right. & 4 & \left. -1 \right] \\ & -1 & \end{array}$$

- Grid boxes: [(-3,1), (-1,2)] [(0,1), (2,4)]

- Data Space: grid boxes + ghost layers: [(-4,0), (0,3)] , [(-1,0), (3,5)]



- Data stored



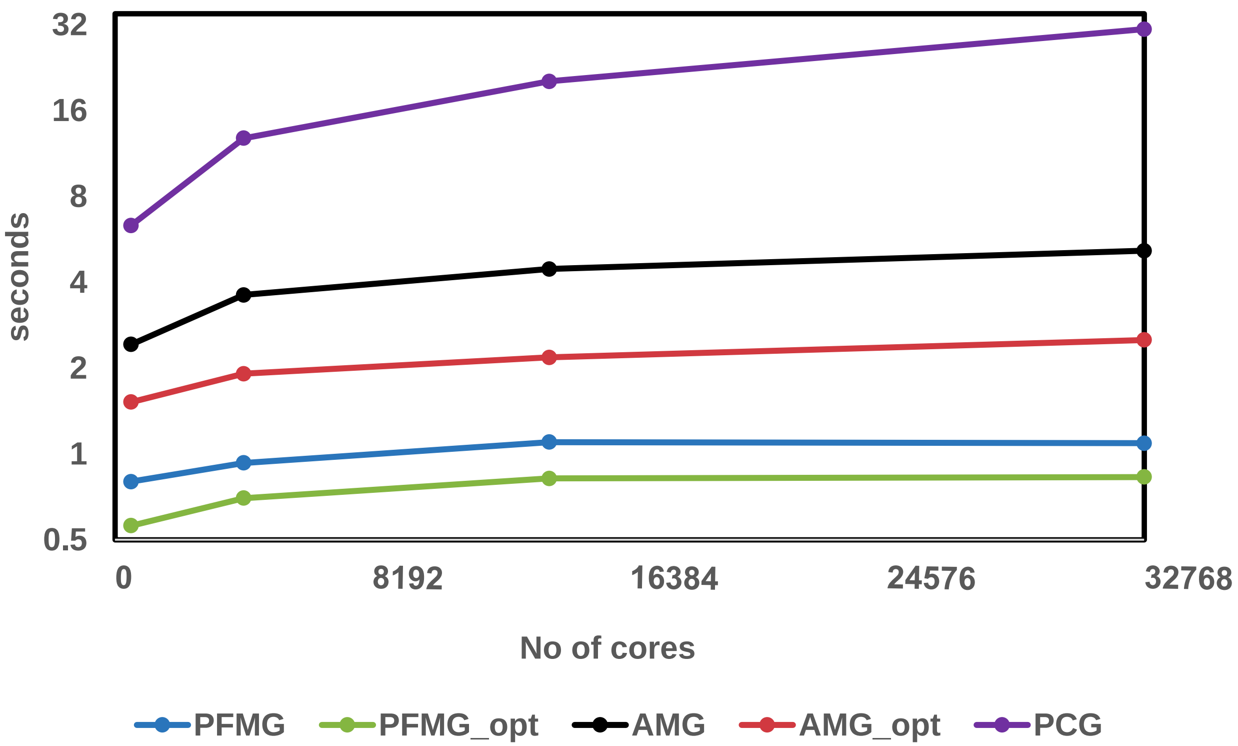
- **Operations applied to stencil entries per box (corresponds to matrix (off) diagonals from a matrix point of view)**

FASTMATH

ECP EXASCALE COMPUTING PROJECT

Algebraic multigrid as preconditioner

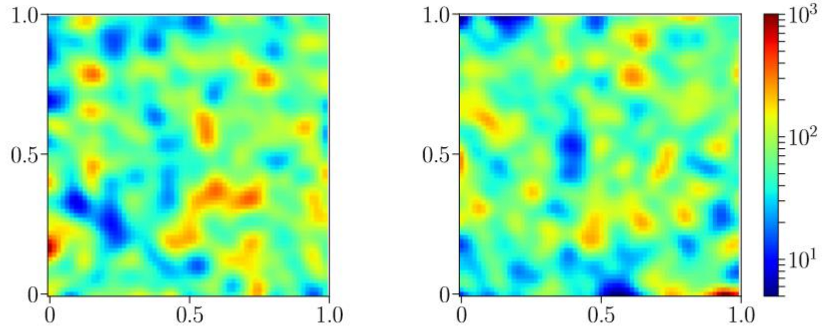
- Generally algebraic multigrid methods are used as preconditioners to Krylov methods, such as conjugate gradient (CG) or GMRES
- This often leads to additional performance improvements



Classic porous media diffusion problem:

$$-\nabla \cdot \kappa \nabla \mathrm{u} = \mathrm{f}$$

with κ having jumps of 2-3 orders of magnitude



Weak scaling: 32x32x32 grid points per core, BG/Q

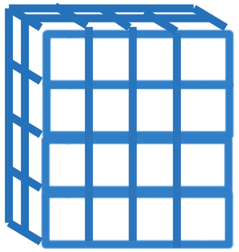
Hands-on Exercises: Algebraic multigrid methods

- Go to https://xsdk-project.github.io/MathPackagesTraining2020/lessons/krylov_amg_hypre/

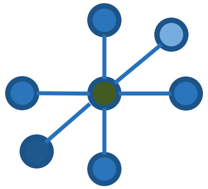
- Poisson equation: $-\Delta\varphi = \text{RHS}$

  with Dirichlet boundary conditions $\varphi = 0$

- Grid: cube



- Finite difference discretization:
  - Central differences for diffusion term
  - 7-point stencil



EXASCALE COMPUTING PROJECT

Thank you!

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC.