Multigrid on the outside: restructuring time integration and adaptivity

Jed Brown jedbrown@mcs.anl.gov (ANL and CU Boulder)
Collaborators in this work:
Debojyoti Ghosh (ANL), Mark Adams (LBL), Matt Knepley (UChicago)

Berkeley Lab, 2014-01-16

Outline

Fast solvers for Implicit Runge-Kutta
- The memory bandwidth problem
- Implicit Runge-Kutta
- Tensor product algebra

τ-adaptivity and multigrid compression
- Reducing communication and memory bandwidth
- Local recovery and postprocessing

Hardware Arithmetic Intensity

Operation	Arithmetic Intensity (flops/B)
Sparse matrix-vector product	1/6
Dense matrix-vector product	1/4
Unassembled matrix-vector product	≈ 8
High-order residual evaluation	> 5

Processor	Bandwidth (GB/s)	Peak (GF/s)	Balance (F/B)
E5-2680 8-core	38	173	4.5
Magny Cours 16-core	49	281	5.7
Blue Gene/Q node	43	205	4.8
Tesla M2090	120	665	5.5
Kepler K20Xm	160	1310	8.2
Xeon Phi SE10P	161	1060	6.6

Optimizing Sparse Mat-Vec

- Order unknowns so vector reuses cache (Cuthill-McKee)
  - Optimal: $\frac{(2 \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)}+\texttt{sizeof(Int)}}$
  - Usually improves strength of ILU and SOR
- Coalesce indices for adjacent rows (Inodes)
  - Optimal: $\frac{(2 \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)}+\texttt{sizeof(Int)}/i}$
  - Can do block SOR (much stronger than scalar SOR)
  - Default in PETSc, turn off with `-mat_no_inode`
  - Requires ordering unknowns so that fields are interlaced, this is (much) better for memory use anyway
- Use explicit blocking, hold one index per block (BAIJ format)
  - Optimal: $\frac{(2 \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)}+\texttt{sizeof(Int)}/b^2}$
  - Block SOR and factorization
  - Symbolic factorization works with blocks (much cheaper)
  - Very regular memory access, unrolled dense kernels
  - Faster insertion: `MatSetValuesBlocked()`

This is a dead end

- Arithmetic intensity $< 1/4$
- Idea: multiple right hand sides

$$\frac{(2k \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)} + \texttt{sizeof(Int)}}, \quad k \ll \text{avg. nz/row}$$

- Problem: popular algorithms have nested data dependencies
  - Time step
    - Nonlinear solve
      - Krylov solve
        - Preconditioner/sparse matrix
- Cannot parallelize/vectorize these nested loops
- Can we create new algorithms to reorder/fuse loops?
  - Reduce latency-sensitivity for communication
  - Reduce memory bandwidth (reuse matrix)

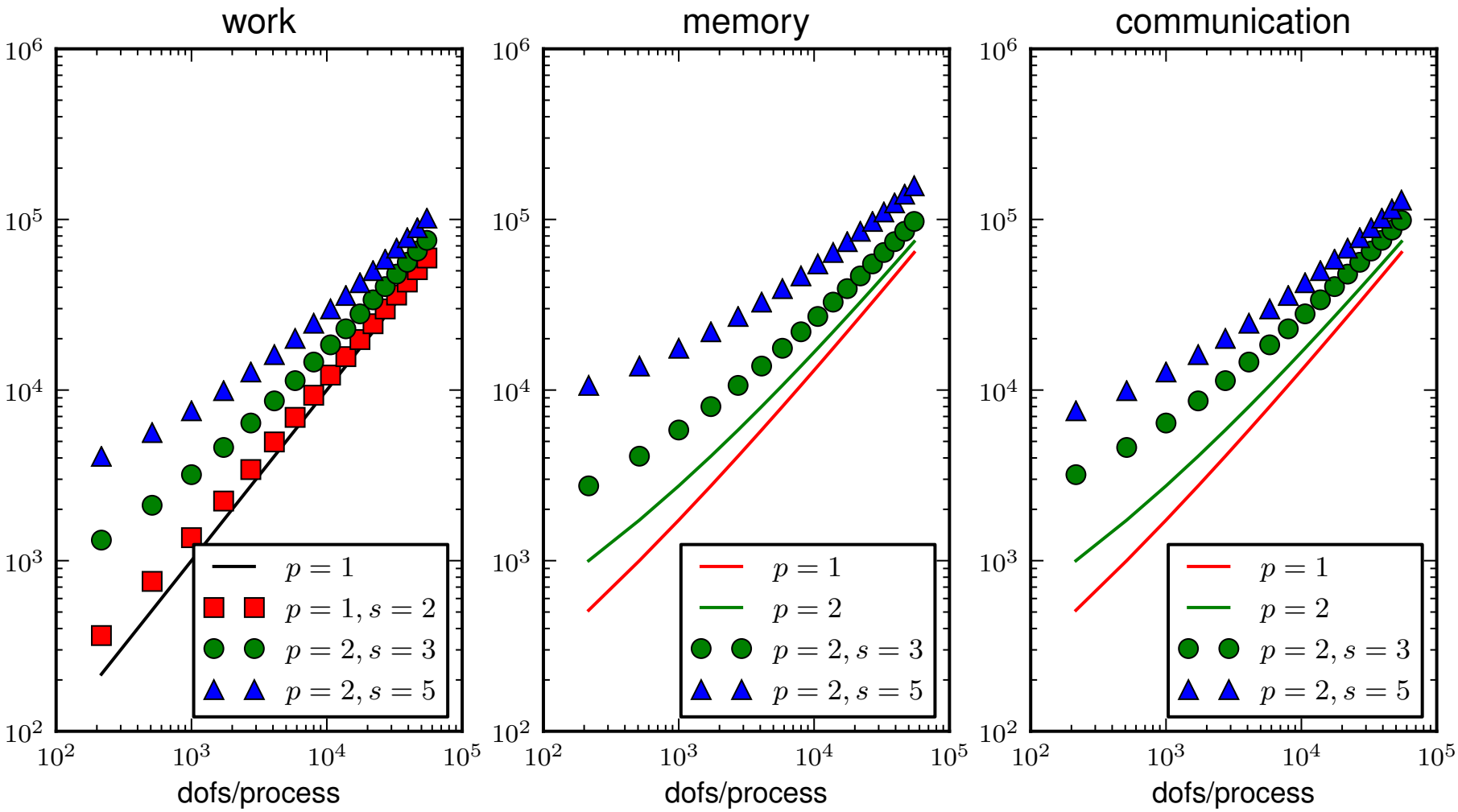
This is a dead end

- Arithmetic intensity $< 1/4$
- Idea: multiple right hand sides

$$\frac{(2k \text{ flops})(\text{bandwidth})}{\texttt{sizeof(Scalar)} + \texttt{sizeof(Int)}}, \quad k \ll \text{avg. nz/row}$$

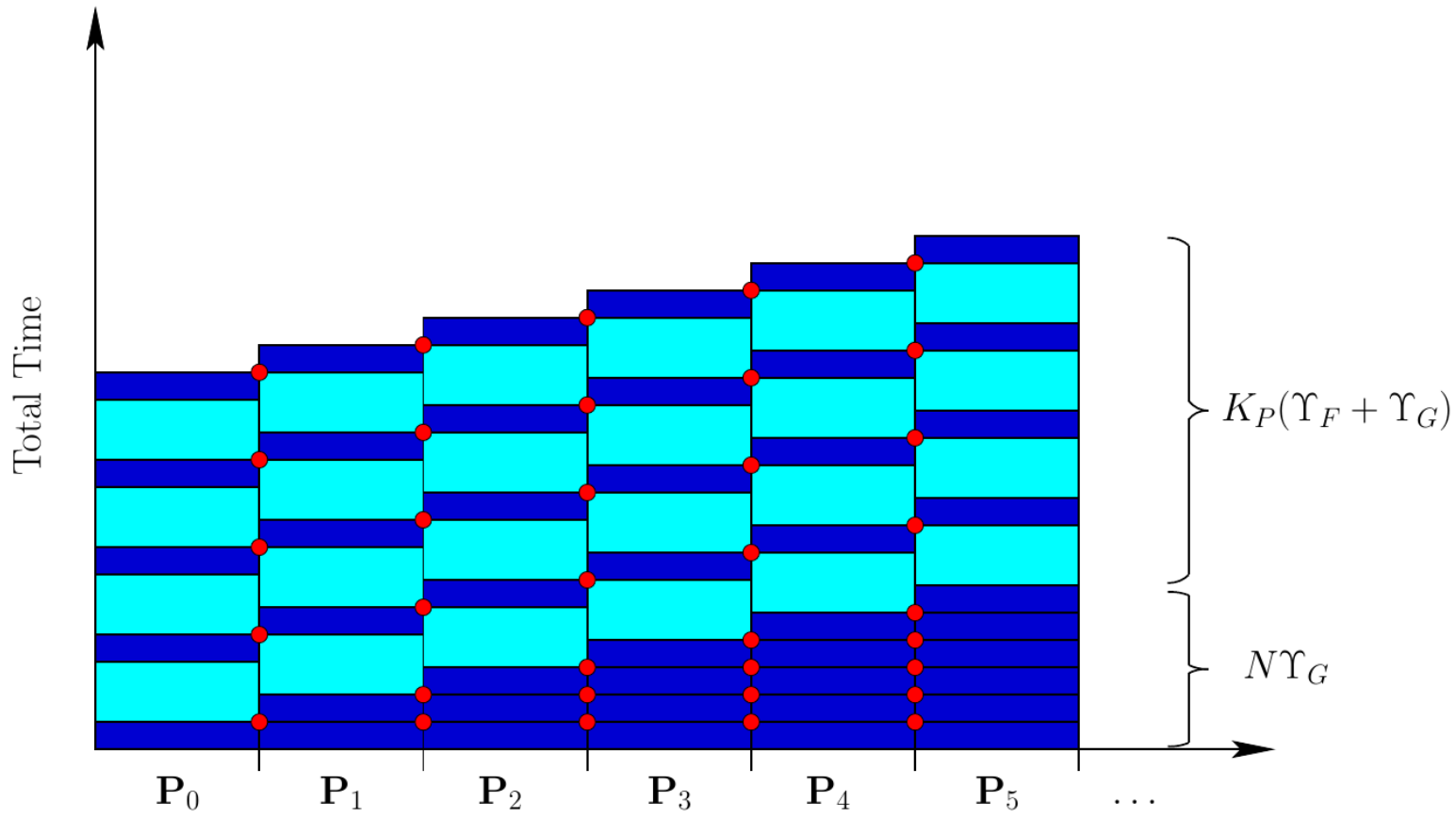
- Problem: popular algorithms have nested data dependencies
  - Time step
    - Nonlinear solve
      - Krylov solve
        - Preconditioner/sparse matrix
- Cannot parallelize/vectorize these nested loops
- Can we create new algorithms to reorder/fuse loops?
  - Reduce latency-sensitivity for communication
  - Reduce memory bandwidth (reuse matrix)

Attempt: *s*-step methods in 3D



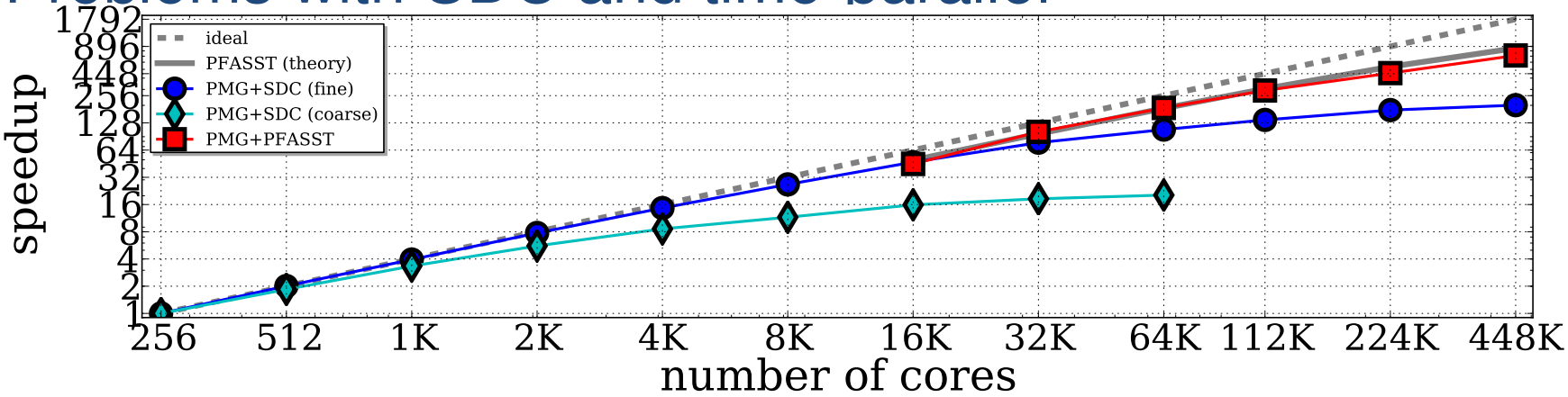
- Amortizing message latency is most important for strong-scaling
- *s*-step methods have high overhead for small subdomains
- Limited choice of preconditioners (none optimal, surface/volume)

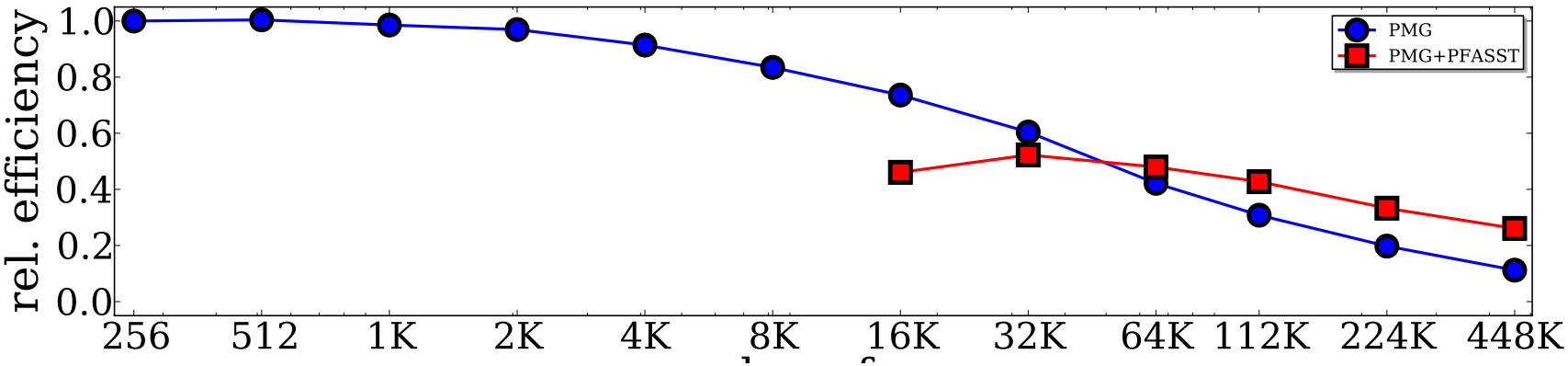
Attempt: space-time methods (multilevel SDC/Parareal)



- PFASST algorithm (Emmett and Minion, 2013)
- Zero-latency messages (cf. performance model of *s*-step)
- Spectral Deferred Correction: iterative, converges to IRK (Gauss, Radau, . . .)
- Stiff problems use implicit basic integrator (synchronizing on spatial

Problems with SDC and time-parallel





c/o Matthew Emmett, parallel compared to sequential SDC

- Number of iterations is not uniform, efficiency starts low
- Arithmetic intensity unchanged
- Parabolic space-time (Greenwald and Brandt/Horton and Vandewalle)

Runge-Kutta methods

$$\dot{u} = F(u)$$

$$\underbrace{\begin{pmatrix} y_1 \\ \vdots \\ y_s \end{pmatrix}}_{Y} = u^n + h \underbrace{\begin{bmatrix} a_{11} & \cdots & a_{1s} \\ \vdots & \ddots & \vdots \\ a_{s1} & \cdots & a_{ss} \end{bmatrix}}_{A} F \begin{pmatrix} y_1 \\ \vdots \\ y_s \end{pmatrix}$$

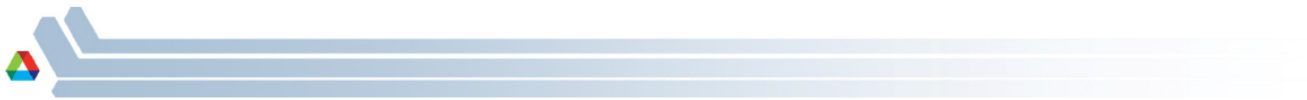
$$u^{n+1} = b^T Y$$

- General framework for one-step methods
- Diagonally implicit: A lower triangular, stage order ≤ 2
- Singly diagonally implicit: all A_{ii} equal, reuse solver setup, stage order ≤ 1
- If A is a general full matrix, all stages are coupled, “implicit RK”

Implicit Runge-Kutta

$$
\begin{array}{c|ccc}
\frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{15} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\
\frac{1}{2} & \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\
\frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} + \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \\
\hline
 & \frac{5}{18} & \frac{4}{9} & \frac{5}{18}
\end{array}
$$

- Implicit Runge-Kutta methods have excellent accuracy and stability properties
- Gauss methods with s stages
  - order $2s$, (s, s) Padé approximation to the exponential
  - A-stable, symplectic
- Radau (IIA) methods with s stages
  - order $2s - 1$, A-stable, L-stable
- Lobatto (IIIC) methods with s stages
  - order $2s - 2$, A-stable, L-stable, self-adjoint
- Stage order s or $s + 1$



Method of Butcher (1976) and Bickart (1977)

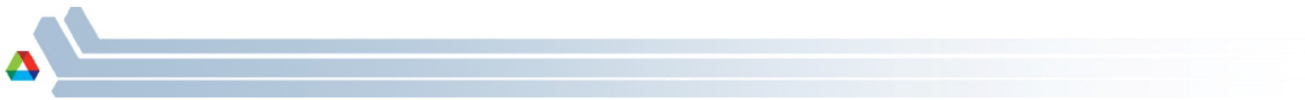
- Newton linearize Runge-Kutta system

$$Y = u^n + hAF(Y)$$

- Solve linear system with tensor product operator

$$S \otimes I_n + I_s \otimes J$$

  where $S = (hA)^{-1}$ is $s \times s$ dense, $J = -\partial F(u)/\partial u$ sparse
- SDC (2000) is Gauss-Seidel with low-order corrector
- Butcher/Bickart method: diagonalize $S = X\Lambda X^{-1}$
  - $\Lambda \otimes I_n + I_s \otimes J$
- s decoupled solves
- Problem: X is exponentially ill-conditioned wrt. s



MatTAIJ: “sparse” tensor product matrices

$$G = I_n \otimes S + J \otimes T$$

- More general than multiple RHS (multivectors)
- Compare to multiple right hand sides in row-major
- Runge-Kutta systems have $T = I_s$ (permuted from Butcher method)
- Stream J through cache once, same efficiency as multiple RHS

Blue Gene/Q test

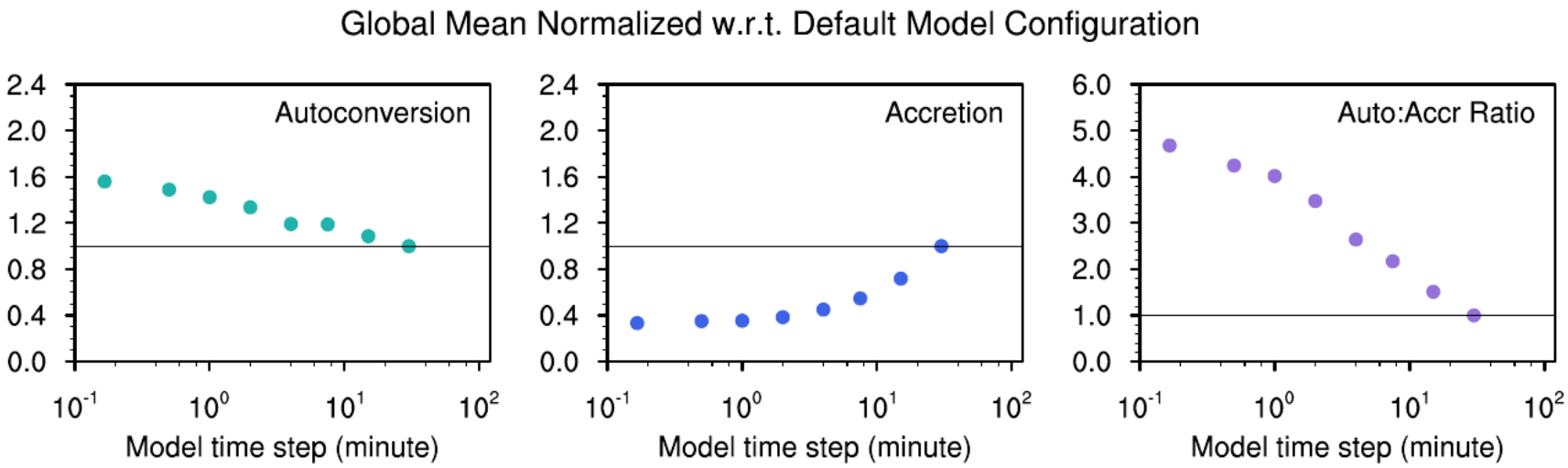
128 nodes, 16 procs/node, small diffusion problem, CG/Jacobi solver

Method	order	nsteps	time
Gauss 4	8	10	3.4345e-01
Gauss 2	4	20	7.6320e-01
Gauss 1	2	40	1.1052e+00

Calibration and accuracy

- Splitting errors plague multi-physics simulation
- Verlet (leapfrog) integration is popular: symplectic and **cheap**
  - Stability problems: damping and even/odd decoupling
- Models calibrated to compensate
  - Force parametrizations in molecular dynamics
  - Atmospheric column physics

Impact of time step on autoconversion vs accretion partitioning (from Hui)



c/o Peter Caldwell (LLNL)

- Models calibrated for “efficient” time step
- Not longer solving the PDEs we write down
- Many FTE-years to recalibrate when discretization changes
- Calibration eats up a big chunk of the IPCC policy timeline

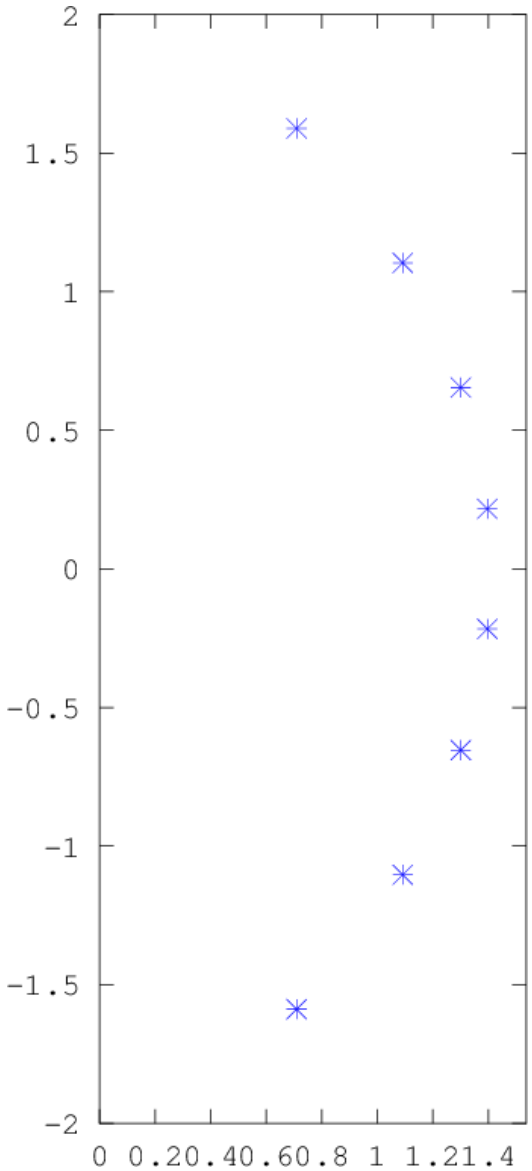
Implicit Runge-Kutta for advection

Table : Total number of iterations (communications or accesses of J) to solve linear advection to $t = 1$ on a 1024-point grid using point-block Jacobi preconditioning of implicit Runge-Kutta matrix. The relative algebraic solver tolerance is 10^{-8}.

Family	Stages	Order	Iterations
Crank-Nicolson/Gauss	1	2	3627
Gauss	2	4	2560
Gauss	4	8	1735
Gauss	8	16	1442

- Naive centered-difference discretization

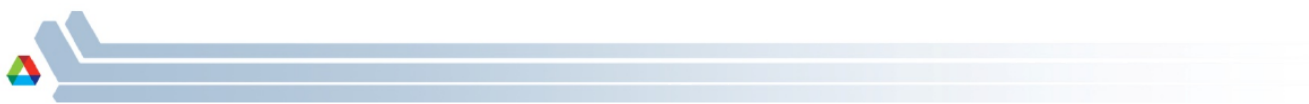
Toward AMG for IRK/tensor-product systems



- Start with $\hat{R} = R \otimes I_s$, $\hat{P} = P \otimes I_s$

$$G_{\text{coarse}} = \hat{R}(I_n \otimes S + J \otimes I_s)\hat{P}$$

- Imaginary component slows convergence
- Idea: incrementally rotate eigenvalues toward real axis on coarse levels
  Enlangga and Nabben *On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian*

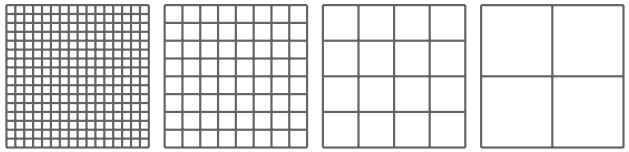


Outline

Fast solvers for Implicit Runge-Kutta
- The memory bandwidth problem
- Implicit Runge-Kutta
- Tensor product algebra

τ-adaptivity and multigrid compression
- Reducing communication and memory bandwidth
- Local recovery and postprocessing

Multigrid Preliminaries



Multigrid is an $O(n)$ method for solving algebraic problems by defining a hierarchy of scale. A multigrid method is constructed from:

1. a series of discretizations
   - coarser approximations of the original problem
   - constructed algebraically or geometrically
2. intergrid transfer operators
   - residual restriction I_h^H (fine to coarse)
   - state restriction $\hat{I}_h^H$ (fine to coarse)
   - partial state interpolation I_H^h (coarse to fine, 'prolongation')
   - state reconstruction $\mathbb{I}_H^h$ (coarse to fine)
3. Smoothers (S)
   - correct the high frequency error components
   - Richardson, Jacobi, Gauss-Seidel, etc.
   - Gauss-Seidel-Newton or optimization methods

τ formulation of Full Approximation Scheme (FAS)

- classical formulation: "coarse grid *accelerates* fine grid ↘↗
- τ formulation: "fine grid feeds back into coarse grid" ↗↘
- To solve $Nu = f$, recursively apply

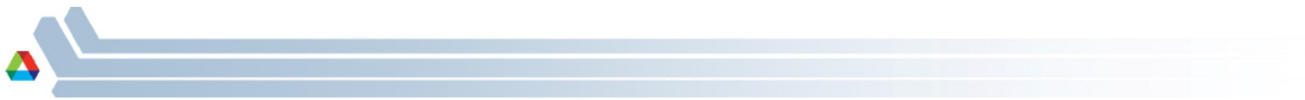
pre-smooth $\quad \tilde{u}^h \leftarrow S^h_{\text{pre}}(u^h_0, f^h)$

solve coarse problem for u^H $\quad N^H u^H = \underbrace{I^H_h f^h}_{f^H} + \underbrace{N^H \hat{I}^H_h \tilde{u}^h - I^H_h N^h \tilde{u}^h}_{\tau^H_h}$

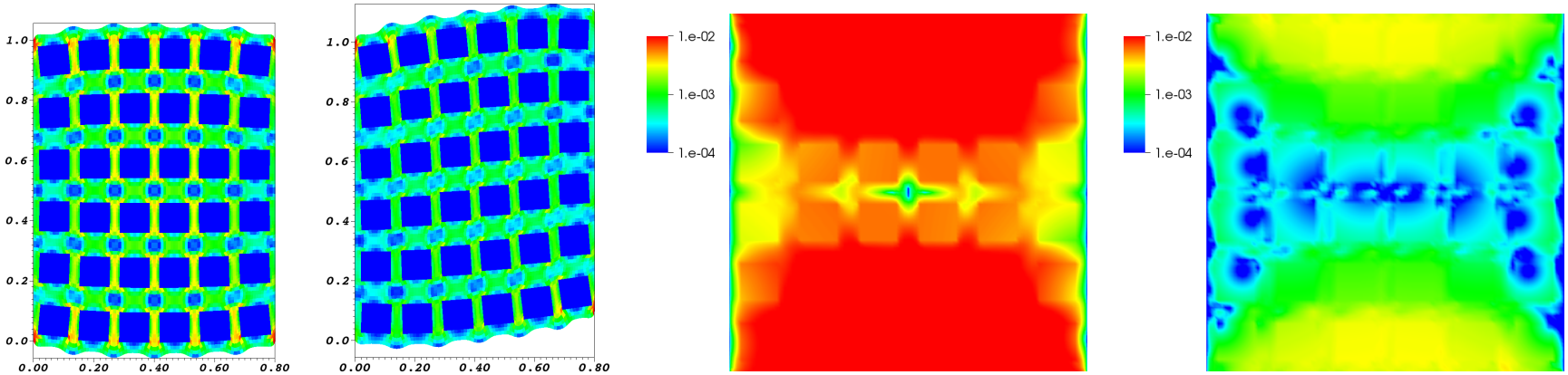
correction and post-smooth $\quad u^h \leftarrow S^h_{\text{post}}\left(\tilde{u}^h + I^h_H(u^H - \hat{I}^H_h \tilde{u}^h), f^h\right)$

I^H_h	residual restriction	$\hat{I}^H_h$	solution restriction
I^h_H	solution interpolation	$f^H = I^H_h f^h$	restricted forcing
$\{S^h_{\text{pre}}, S^h_{\text{post}}\}$	smoothing operations on the fine grid		

- At convergence, $u^{H*} = \hat{I}^H_h u^{h*}$ solves the τ-corrected coarse grid equation $N^H u^H = f^H + \tau^H_h$, thus τ^H_h is the "fine grid feedback" that makes the coarse grid equation accurate.
- τ^H_h is *local* and need only be recomputed where it becomes stale.



τ corrections

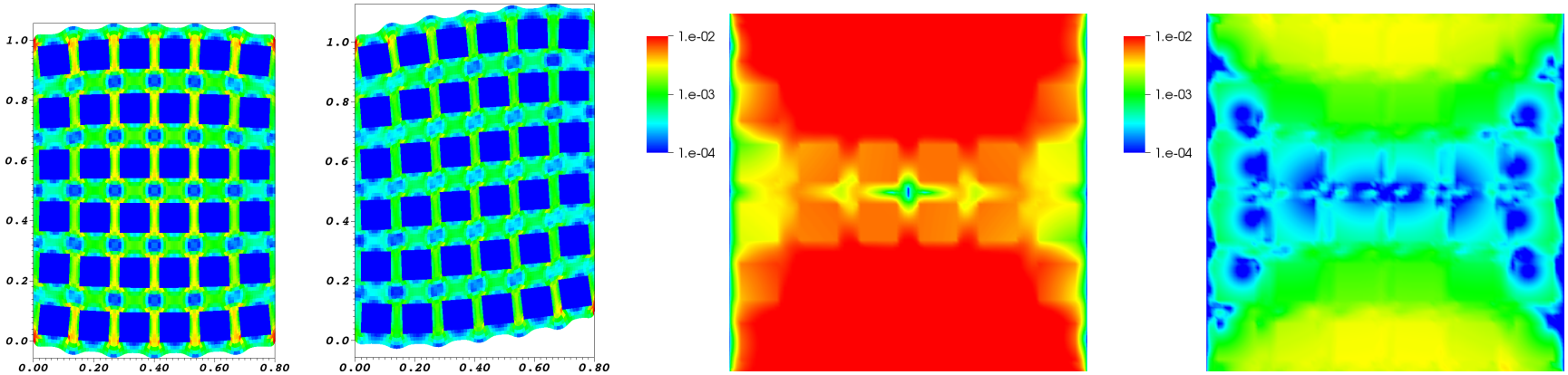


- Plane strain elasticity, $E = 1000, \nu = 0.4$ inclusions in $E = 1, \nu = 0.2$ material, coarsen by 3^2.
- Solve initial problem everywhere and compute $\tau_h^H = A^H \hat{I}_h^H u^h - I_h^H A^h u^h$
- Change boundary conditions and solve FAS coarse problem

$$N^H \acute{u}^H = \underbrace{I_h^H \acute{f}^h}_{\acute{f}^H} + \underbrace{N^H \hat{I}_h^H \tilde{u}^h - I_h^H N^h \tilde{u}^h}_{\tau_h^H}$$

- Prolong, post-smooth, compute error $e^h = \acute{u}^h - (N^h)^{-1} \acute{f}^h$
- Coarse grid *with* τ is nearly $10\times$ better accuracy

τ corrections



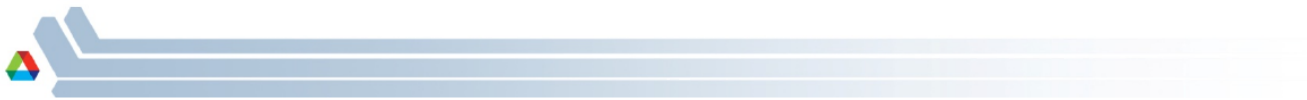
- Plane strain elasticity, $E = 1000, \nu = 0.4$ inclusions in $E = 1, \nu = 0.2$ material, coarsen by 3^2.
- Solve initial problem everywhere and compute $\tau_h^H = A^H \hat{I}_h^H u^h - I_h^H A^h u^h$
- Change boundary conditions and solve FAS coarse problem

$$N^H \acute{u}^H = \underbrace{I_h^H \acute{f}^h}_{\acute{f}^H} + \underbrace{N^H \hat{I}_h^H \tilde{u}^h - I_h^H N^h \tilde{u}^h}_{\tau_h^H}$$

- Prolong, post-smooth, compute error $e^h = \acute{u}^h - (N^h)^{-1} \acute{f}^h$
- Coarse grid *with* τ is nearly $10\times$ better accuracy

τ adaptivity for heterogeneous media

- Applications
  - Geo: reservoir engineering, lithosphere dynamics (subduction, rupture/fault dynamics)
  - carbon fiber, biological tissues, fracture
  - Conventional adaptivity fails
- Traditional adaptive methods fail
  - Solutions are not “smooth”
  - Cannot build accurate coarse space without scale separation
- τ adaptivity
  - Fine-grid work needed everywhere at first
  - Then τ becomes accurate in nearly-linear regions
  - Only visit fine grids in “interesting” places: active nonlinearity, drastic change of solution

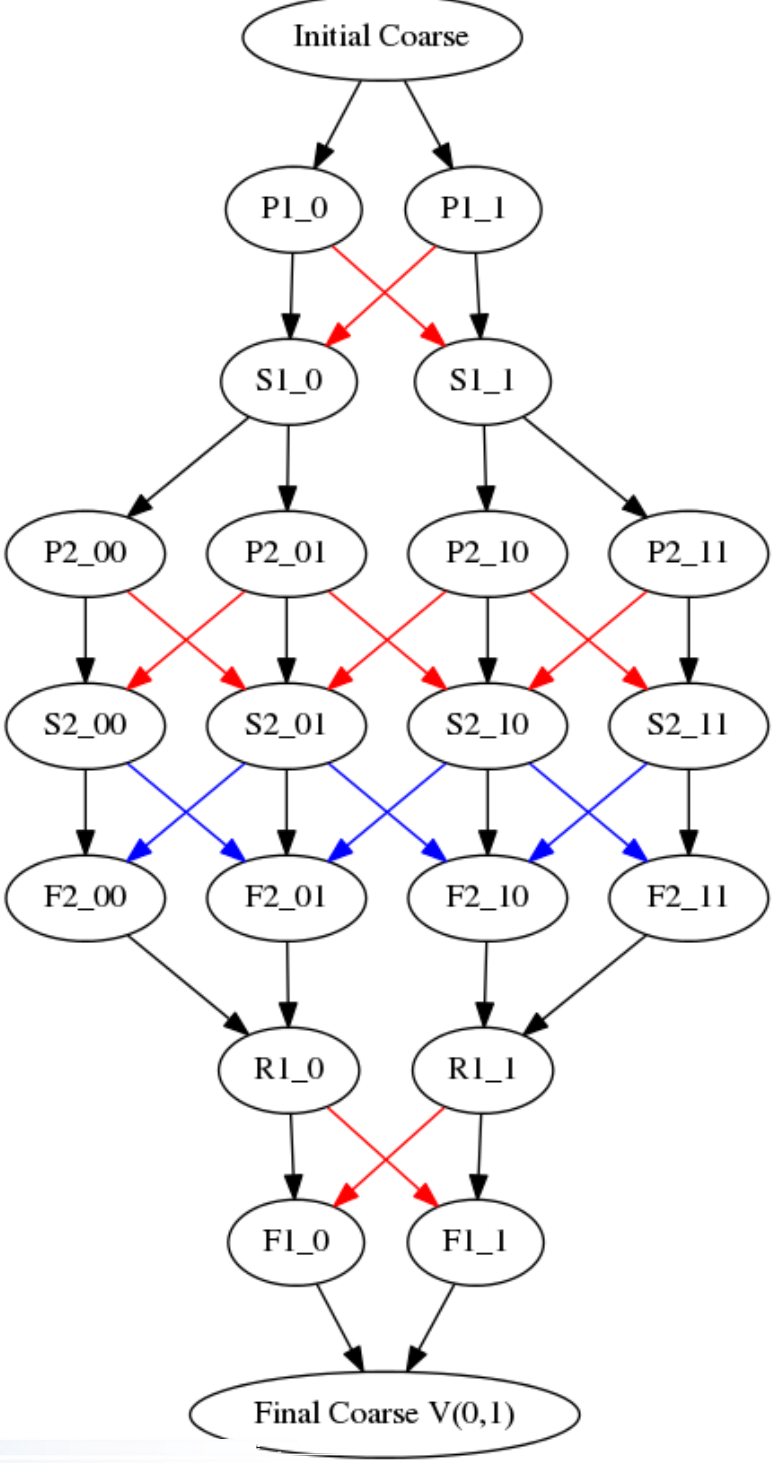


Comparison to nonlinear domain decomposition

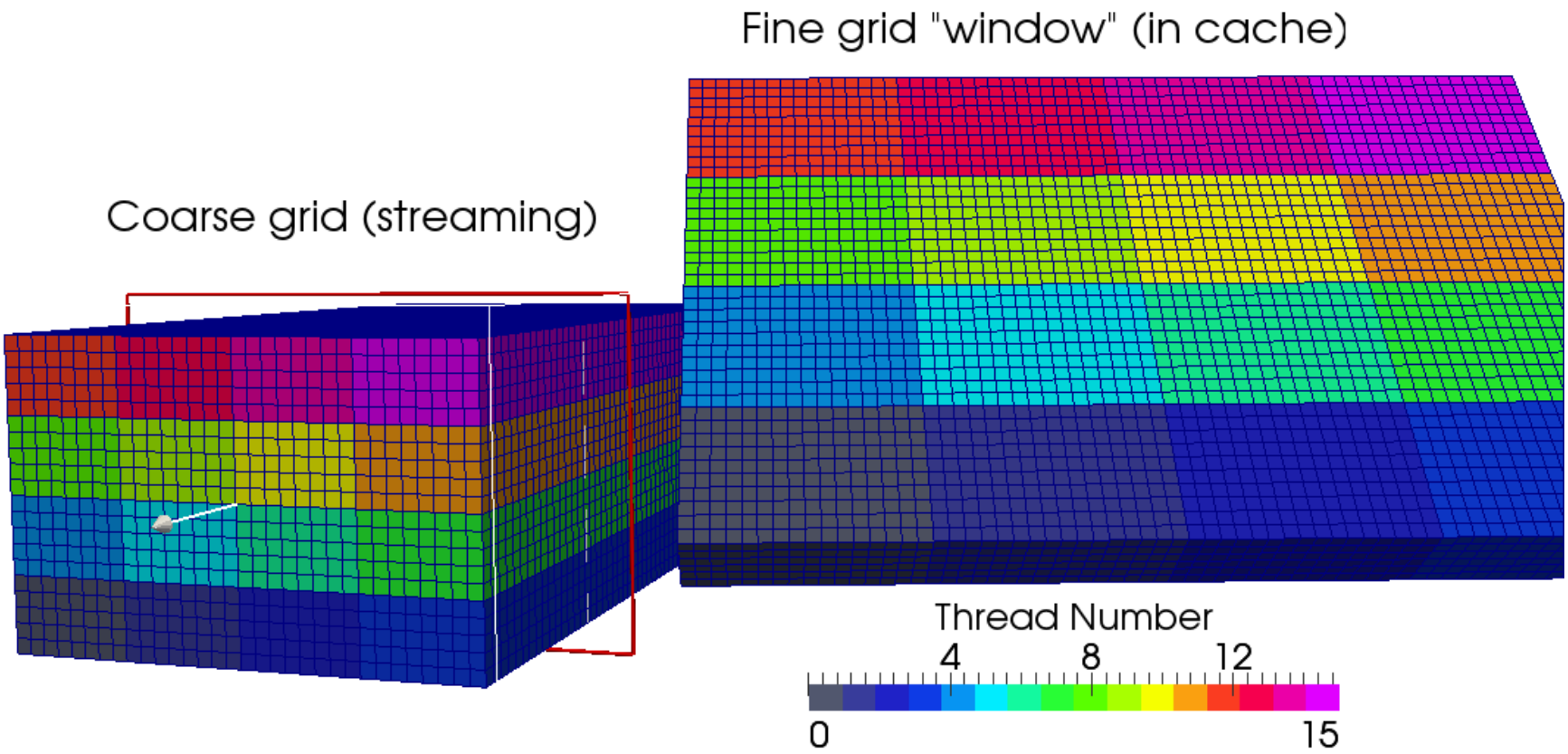
- ASPIN (Additive Schwarz preconditioned inexact Newton)
  - Cai and Keyes (2003)
  - More local iterations in strongly nonlinear regions
  - Each nonlinear iteration only propagates information locally
  - Many real nonlinearities are activated by long-range forces
    - locking in granular media (gravel, granola)
    - binding in steel fittings, crack propagation
  - Two-stage algorithm has different load balancing
    - Nonlinear subdomain solves
    - Global linear solve
- τ adaptivity
  - Minimum effort to communicate long-range information
  - Nonlinearity sees effects as accurate as with global fine-grid feedback
  - Fine-grid work always proportional to “interesting” changes

Low communication MG

- red arrows can be removed by τ-FAS with overlap
- blue arrows can also be removed, but then algebraic convergence stalls when discretization error is reached
- no simple way to check that discretization error is obtained
- if fine grid state is not stored, use compatible relaxation to complete prolongation P



Reducing memory bandwidth



- Sweep through “coarse” grid with moving window
- Zoom in on new slab, construct fine grid “window” in-cache
- Interpolate to new fine grid, apply pipelined smoother (*s*-step)
- Compute residual, accumulate restriction of state and residual into coarse grid, expire slab from window

Arithmetic intensity of sweeping visit

- Assume 3D cell-centered, 7-point stencil
- 14 flops/cell for second order interpolation
- $\geq$ 15 flops/cell for fine-grid residual or point smoother
- 2 flops/cell to enforce coarse-grid compatibility
- 2 flops/cell for plane restriction
- assume coarse grid points are reused in cache
- Fused visit reads u^H and writes $\hat{I}_h^H u^h$ and $I_h^H r^h$
- Arithmetic Intensity

$$\frac{\overbrace{15}^{\text{interp}} + \overbrace{2\cdot(15+2)}^{\text{compatible relaxation}} + \overbrace{2\cdot 15}^{\text{smooth}} + \overbrace{15}^{\text{residual}} + \overbrace{2}^{\text{restrict}}}{3\cdot \texttt{sizeof(scalar)} / \underbrace{2^3}_{\text{coarsening}}} \gtrsim 30 \qquad (1)$$

- Still $\gtrsim$ 10 with non-compressible fine-grid forcing

Regularity

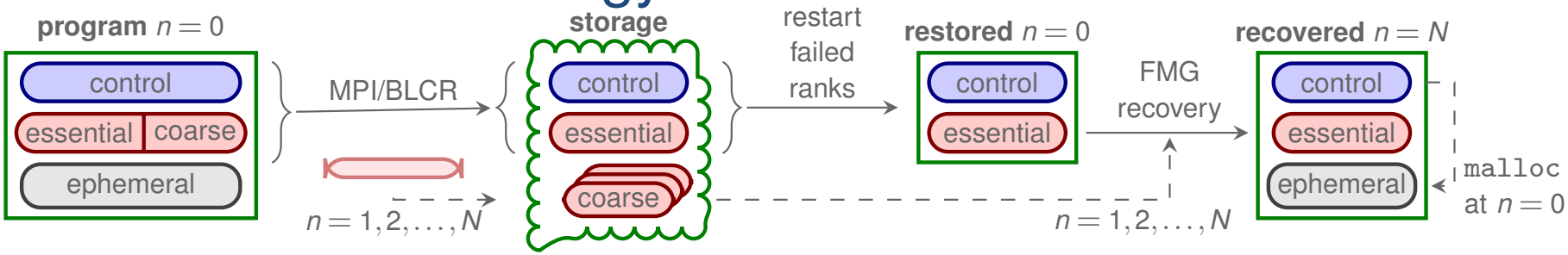
Accuracy of recovery depends on operator regularity

- Even with regularity, we can only converge up to discretization error, unless we add a *consistent* fine-grid residual evaluation
- Visit fine grid with some overlap, but patches do not agree exactly in overlap
- Need decay length for high-frequency error components (those that restrict to zero) that is bounded with respect to grid size
- Required overlap J is proportional to the number of cells to cover decay length
- Can enrich coarse space along boundary, but causes loss of coarse-grid sparsity
- Brandt and Diskin (1994) has two-grid LFA showing $J \lesssim 2$ is sufficient for Laplacian
- With L levels, overlap $J(k)$ on level k,

$$2J(k) \geq s(L-k+1)$$

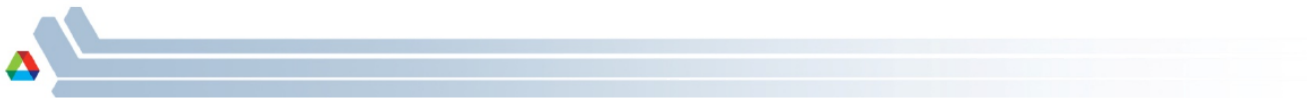
where s is the smoothness order of the solution or the discretization order (whichever is smaller)

Basic resilience strategy

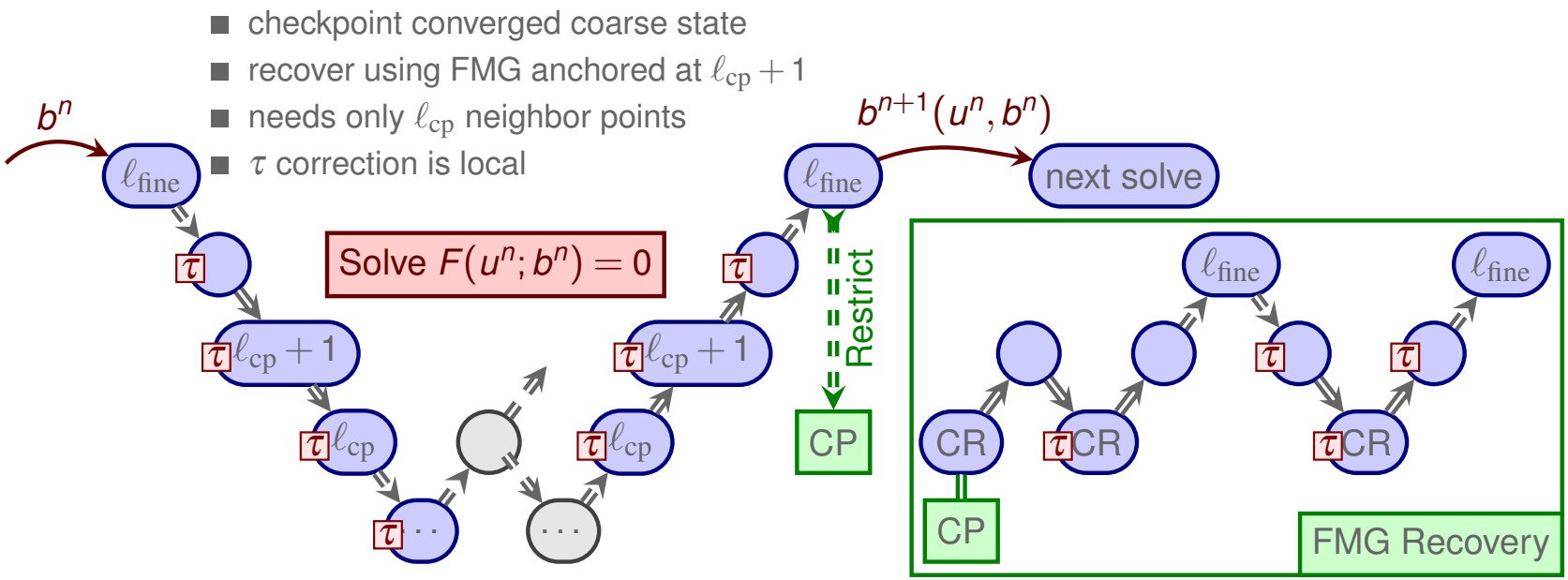


control	contains program stack, solver configuration, etc.
essential	program state that cannot be easily reconstructed: time-dependent solution, current optimization/bifurcation iterate
ephemeral	easily recovered structures: assembled matrices, preconditioners, residuals, Runge-Kutta stage solutions

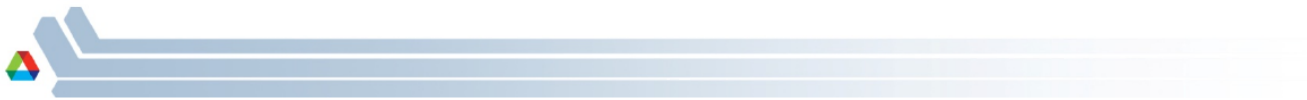
- Essential state at time/optimization step n is inherently globally coupled to step $n-1$ (otherwise we could use an explicit method)
- *Coarse* level checkpoints are orders of magnitude smaller, but allow rapid recovery of essential state
- FMG recovery needs only nearest neighbors



Multiscale compression and recovery using τ form



- Normal multigrid cycles visit all levels moving from $n \rightarrow n+1$
- FMG recovery only accesses levels finer than ℓ_{CP}
- Only failed processes and neighbors participate in recovery
- Lightweight checkpointing for transient adjoint computation
- Postprocessing applications, e.g., in-situ visualization at high temporal resolution in part of the domain



First-order cost model for FAS resilience

Extend first-order locality-unaware model of Young (1974):

- t_W time to write a heavy fine-grid checkpointed state
- t_R time to read back lost state
- R fraction of forward simulation needed for recomputation from a saved state
- P the heavy checkpoint interval
- M mean time to failure

Neglect cost of I/O for lightweight coarse-grid checkpoints

$$\text{Overhead} = 1 - \text{AppUtilization} = \underbrace{\frac{t_W}{P}}_{\text{writing}} + \underbrace{\frac{t_R}{M}}_{\text{reading after failure}} + \underbrace{\frac{RP}{2M}}_{\text{recomputation}}$$

Minimized for a heavy checkpointing interval $P = \sqrt{2Mt_W/R}$

$$\text{Overhead}^* = \sqrt{2t_W R/M} + t_R/M$$

where the first term is always larger than the second. Conventional checkpointing schemes store only fine-grid state, thus $R = 1$ (recovery costs the same as initial computation).

Outlook

- Implicit Runge-Kutta
  - Working: Tensor product matrices (TAIJ format) and one-level methods
  - Next up: Algebraic multigrid for tensor product operators
  - Research: imaginary rotation in coarse operators (cf. MG for Helmholtz)
  - Stochastic Galerkin has same structure
  - Is it possible to design methods with well-conditioned $S = X\Lambda X^{-1}$
- τ-adaptivity
  - nonlinear smoothers (and discretizations)
  - dynamic load balancing
  - reliability of error estimates for refreshing τ