


Extreme-scale Multigrid in Space and Time

2015 SIAM Conference on Computational Science and Engineering
Salt Lake City, March 14-18, 2015

Robert D. Falgout
Center for Applied Scientific Computing

 Lawrence Livermore
National Laboratory

LLNL PRES 668413

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.



Outline – two main parts

- Multigrid (in space)
 - Motivation and background
 - Reducing communication in parallel multigrid

- Multigrid in time
 - Motivation and basic approach
 - MGRIT – multigrid reduction (MGR) in time
 - Parallel open source code XBraid
 - Brief literature survey

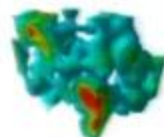
- Thanks, summary and conclusions

Multigrid will play an important role for addressing exascale challenges

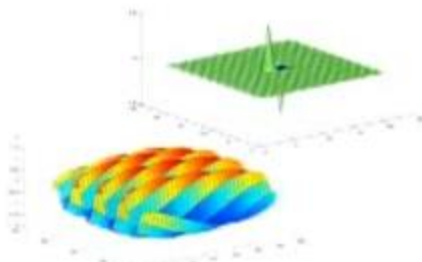
- For many applications, the fastest and most scalable solvers are multigrid methods
- Exascale solver algorithms will need to:
 - Exhibit extreme levels of parallelism (**exascale** → 1B cores)
 - Minimize data movement & exploit machine heterogeneity
 - Demonstrate resilience to faults
- **Multilevel methods are ideal**
 - Key feature: Optimal $O(N)$
- **Research challenge:**
 - No optimal solvers yet for some applications, even in serial!
 - Parallel computing increases difficulty



Elasticity / Plasticity

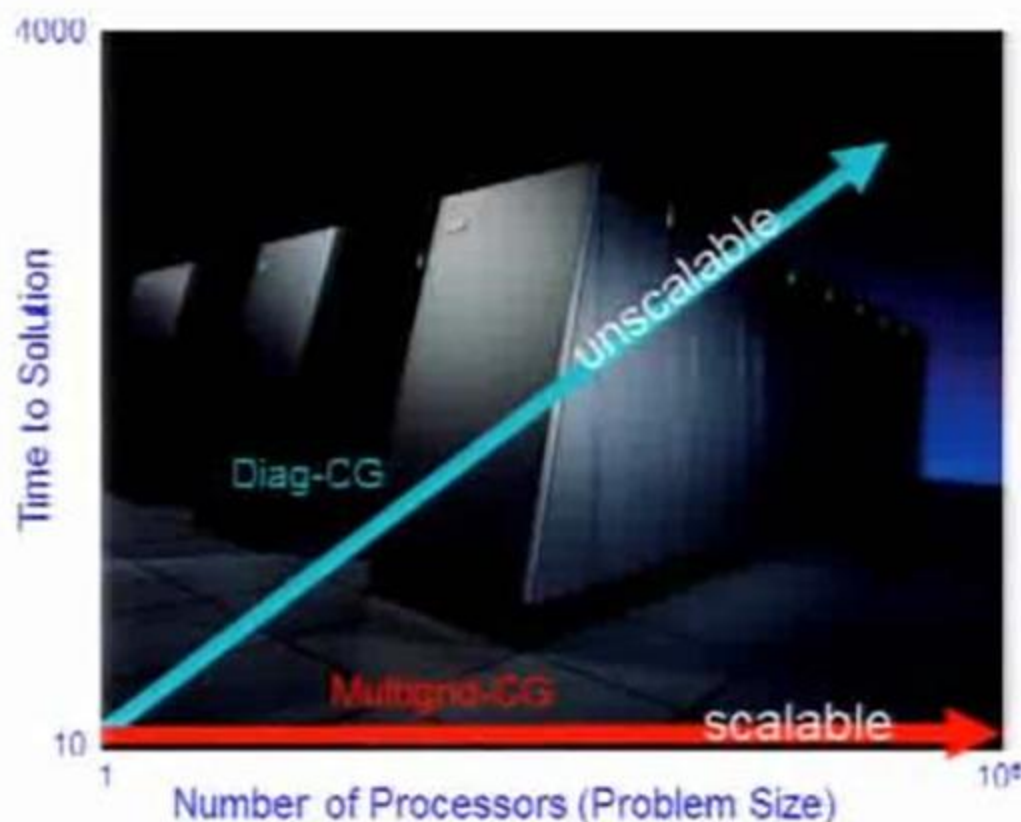


Quantum Chromodynamics



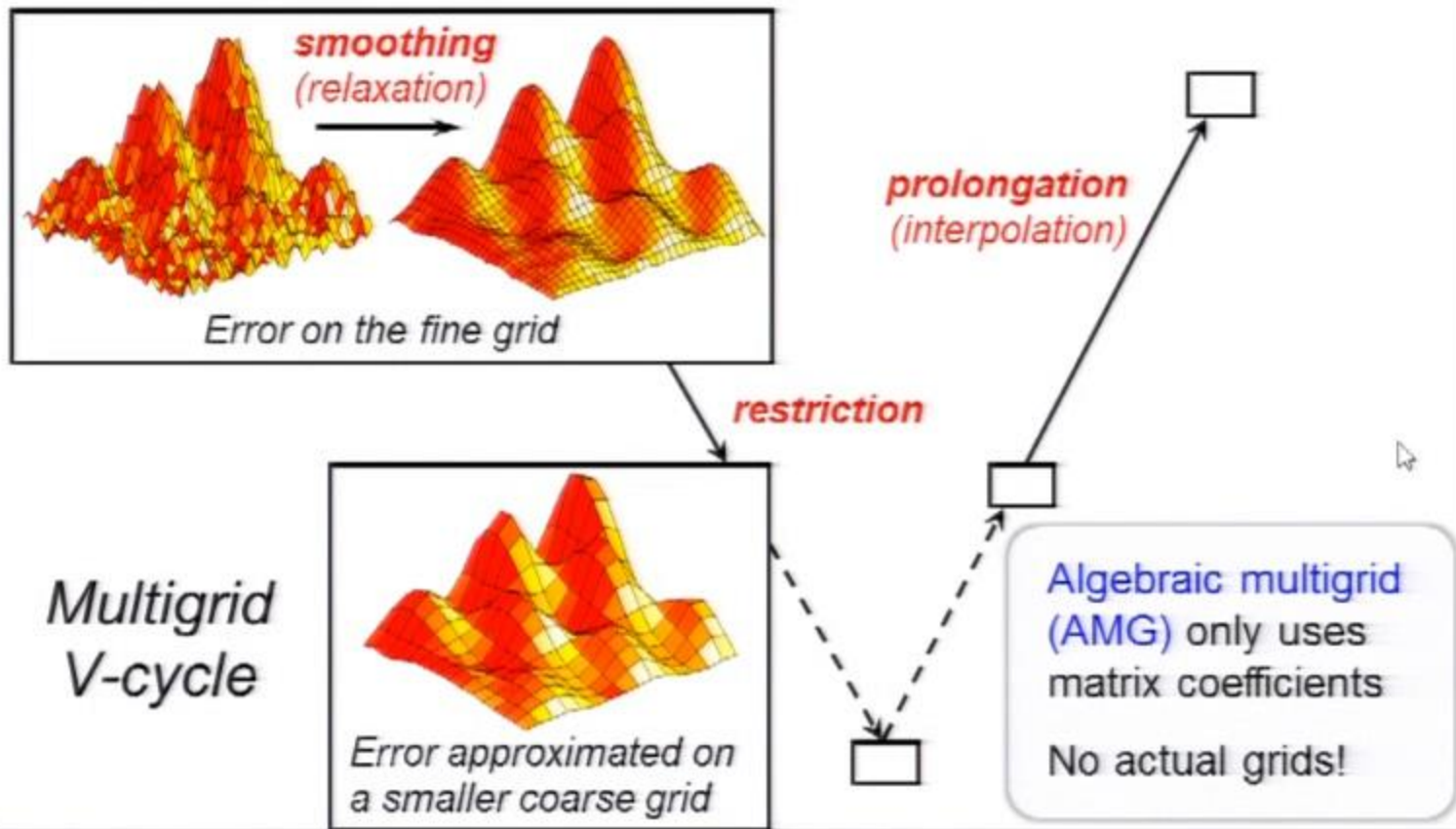
Helmholtz Modes

Multigrid solvers have $O(N)$ complexity, and hence have good scaling potential

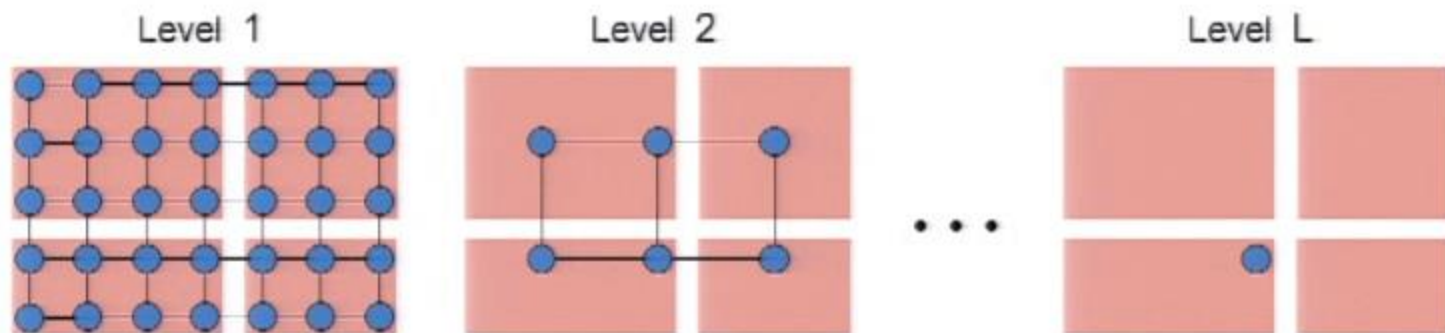


- Weak scaling – want constant solution time as problem size grows in proportion to the number of processors

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



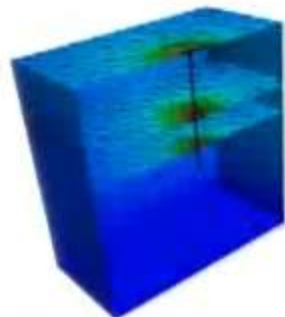
Straightforward MG parallelization yields optimal-order performance for V-cycles



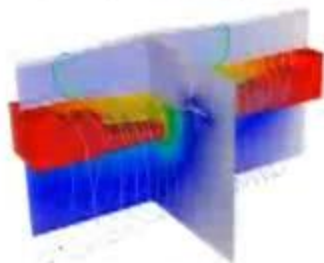
- ~ 1.5 million idle cores on Sequoia!
- Multigrid has a high degree of concurrency
 - Size of the **sequential component** is **only $O(\log N)$!**
 - This is often the **minimum size achievable**
- Parallel performance model has the expected log term
$$T_V = O(\log N)(\text{comm latency}) + O(\Gamma_p)(\text{comm rate}) + O(\Omega_p)(\text{flop rate})$$

Parallel computing imposes restrictions on multigrid algorithm development

- Avoid sequential techniques
 - Classical AMG coarsening
 - Gauss-Seidel smoother
 - Cycles with large sequential component
 - F-cycle: $O(\log^2 N)$
 - W-cycle: $O(2^{\log N}) = O(N)$
- Control communication
 - Galerkin coarse-grid operators (P^TAP) can lead to high communication costs in AMG
- **Need both CS and Math advances!**
 - New methods have new convergence and robustness characteristics
 - Successful addressing issues so far

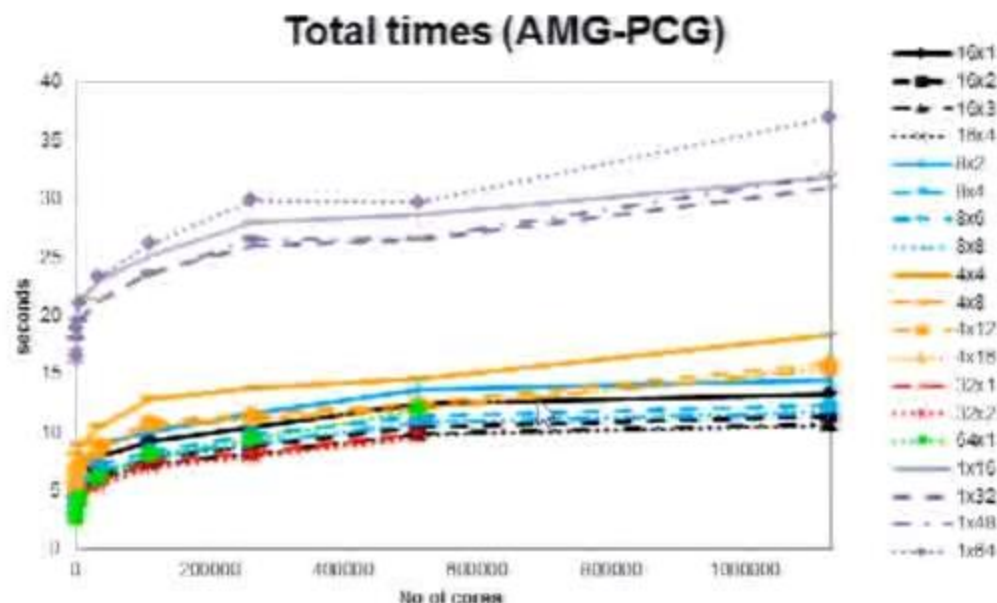


10x speedup for subsurface problems with new coarsening and interpolation approach



Magnetic flux compression generator simulation enabled by MG smoother research

Parallel AMG in *hypre* now scales to 1.1M cores on Sequoia (IBM BG/Q)

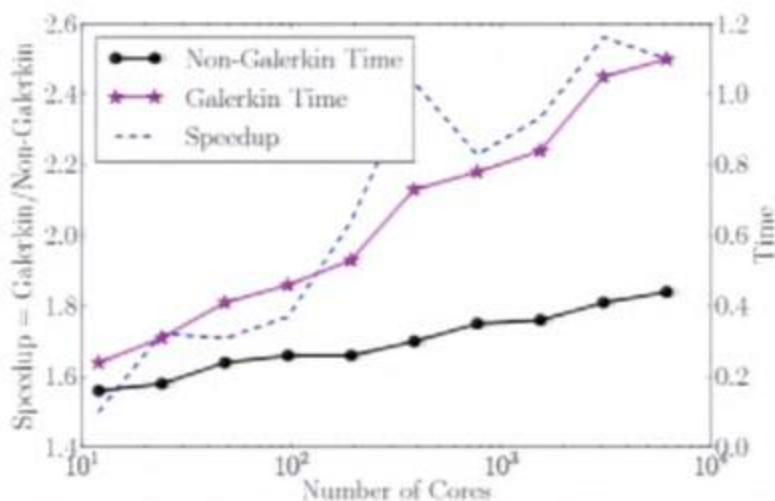


hypre

- $m \times n$ denotes m MPI tasks and n OpenMP threads per node
- Largest problem above: 72B unknowns on 1.1M cores

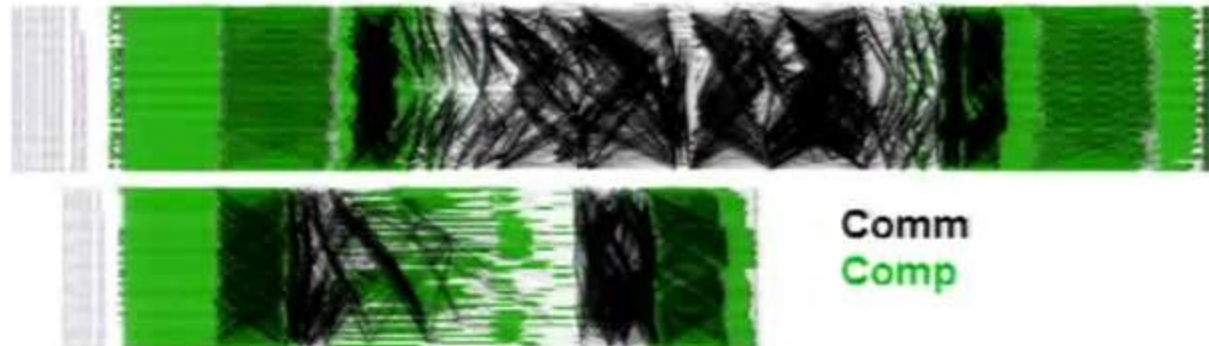
Current spatial-MG research focuses on reducing parallel communication costs (1)

- Non-Galerkin AMG replaces the usual coarse-grid operators with sparser ones
 - Speedups from 1.2x - 2.4x over existing AMG
 - In *hypra* 2.10.0b



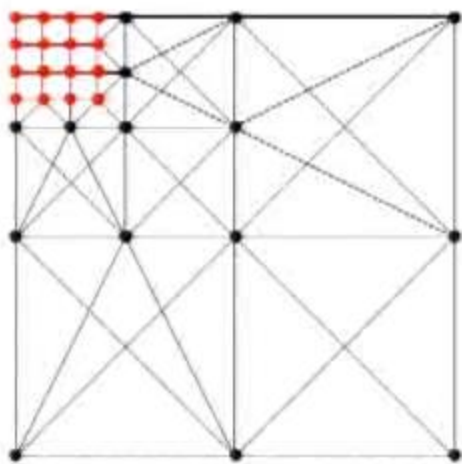
Current spatial-MG research focuses on reducing parallel communication costs (2)

- **Multi-additive AMG** exploits a theoretical identity to inherit the parallelization benefits of additive methods and the convergence properties of multiplicative
 - Additive MG is good at overlapping communication and computation, but converges slower than multiplicative MG
 - **Speedups of 2x** over existing AMG
 - In *hypra* 2.10.0b



Current spatial-MG research focuses on reducing parallel communication costs (3)

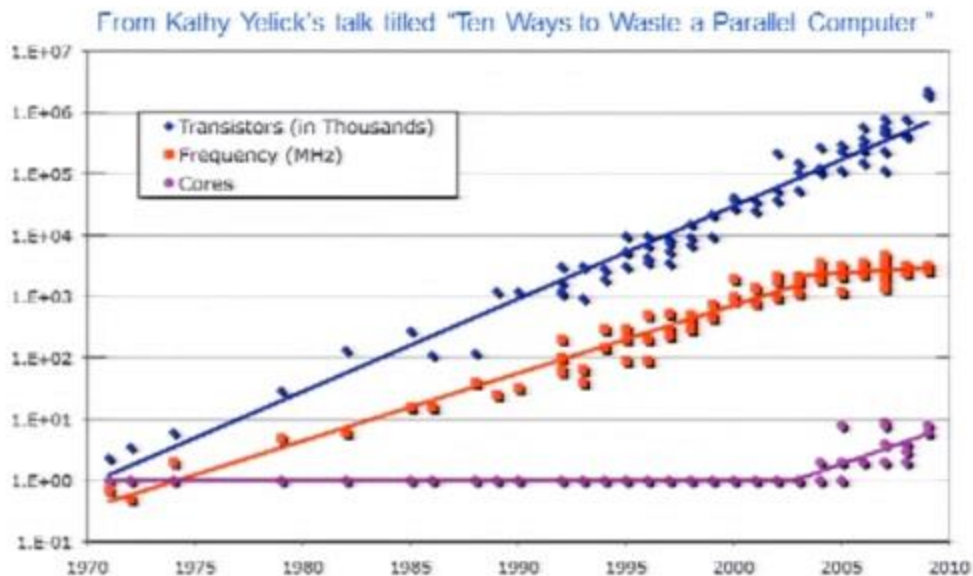
- AMG domain decomposition (AMG-DD) employs cheap global problems to speed up convergence
 - Constructs problems algebraically from an existing method
 - Potential for FMG convergence with only $\log N$ latency!
 - Implementing parallel code



Parallel Time Integration



Traditional time integration will become a sequential bottleneck



- Clock rates are no longer increasing – faster speed is now achieved through more concurrency
- **Parallel time integration methods are needed!**

How on Earth can this be possible?



You can't solve at a given time point until you know the solution at the previous point ... and you can't compute the previous point until you know the one before that ... etc.

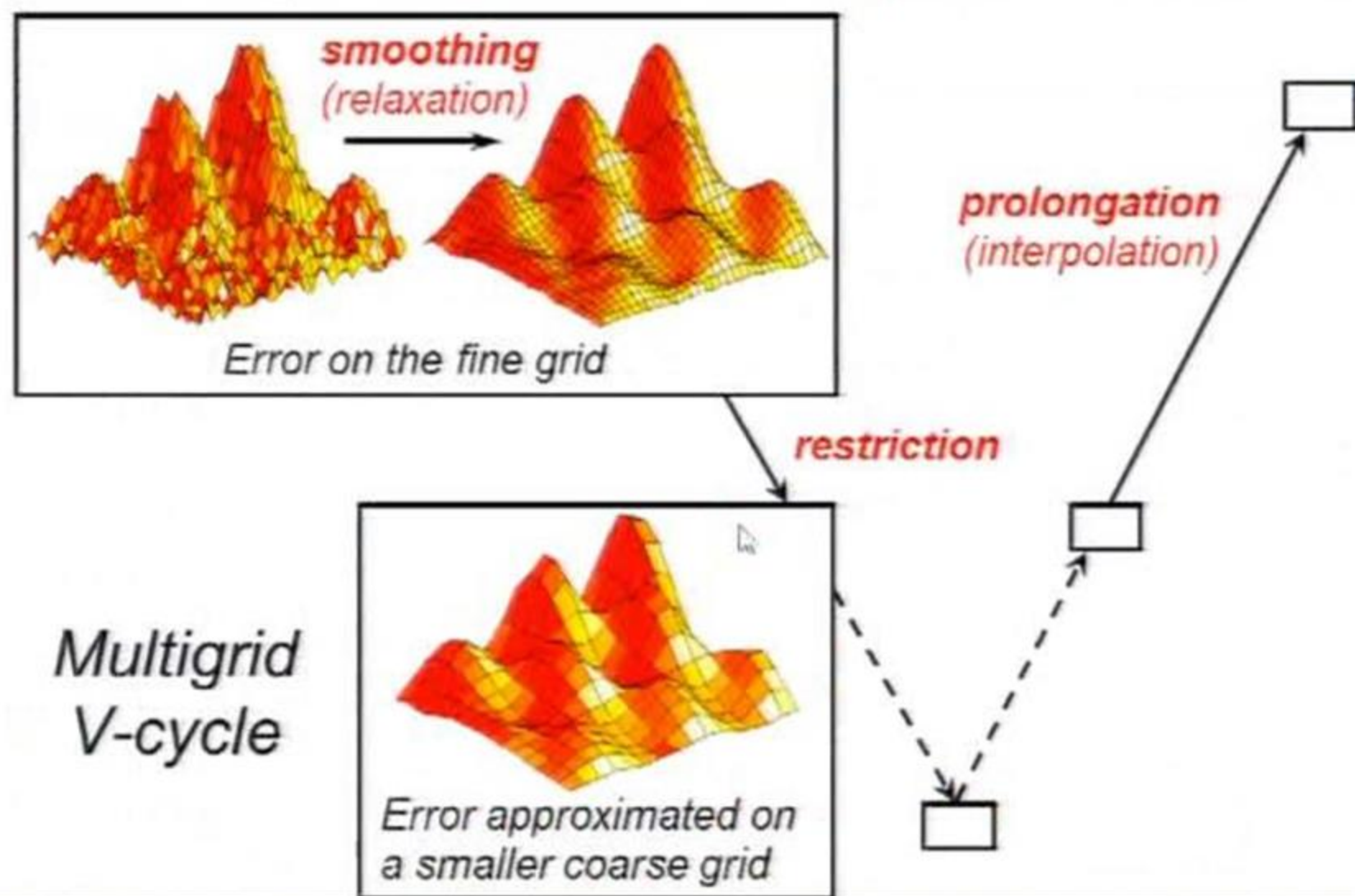
- Is this really any different from space (e.g., diffusion)?



... and the dependence is in both directions (is this harder or easier?)

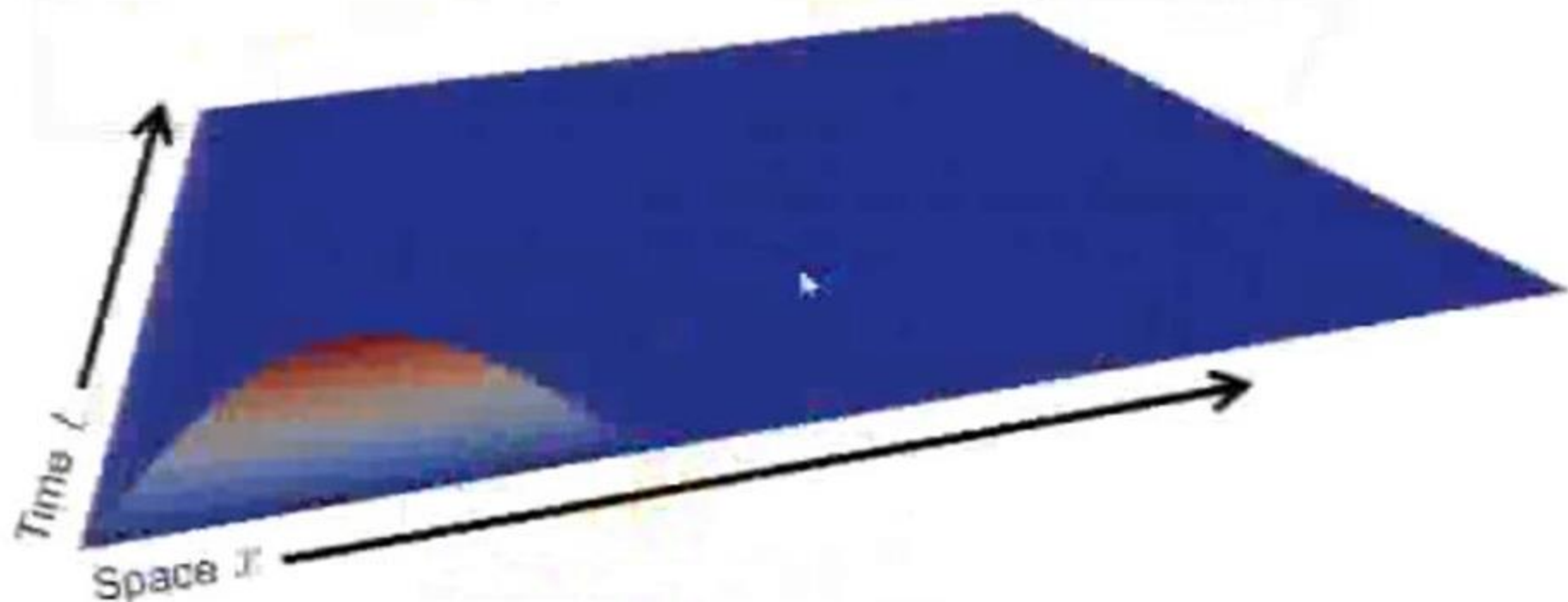
- MG is used routinely to solve the spatial problem
- Why not use it to solve the time problem too?

One approach for parallel-in-time: leverage spatial multigrid research



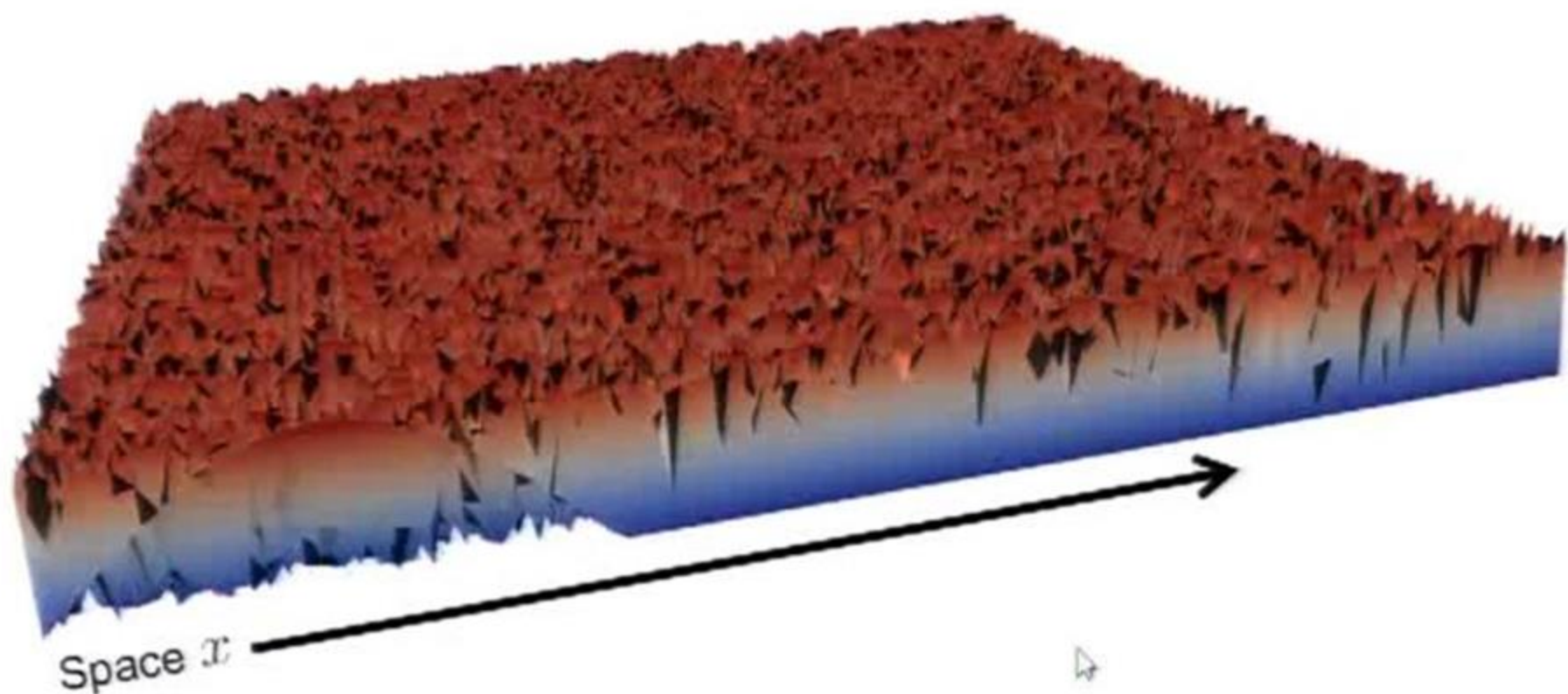
Time stepping is sequential

- Simple advection equation, $u_t = -cu_x$
- Initial condition is a wave



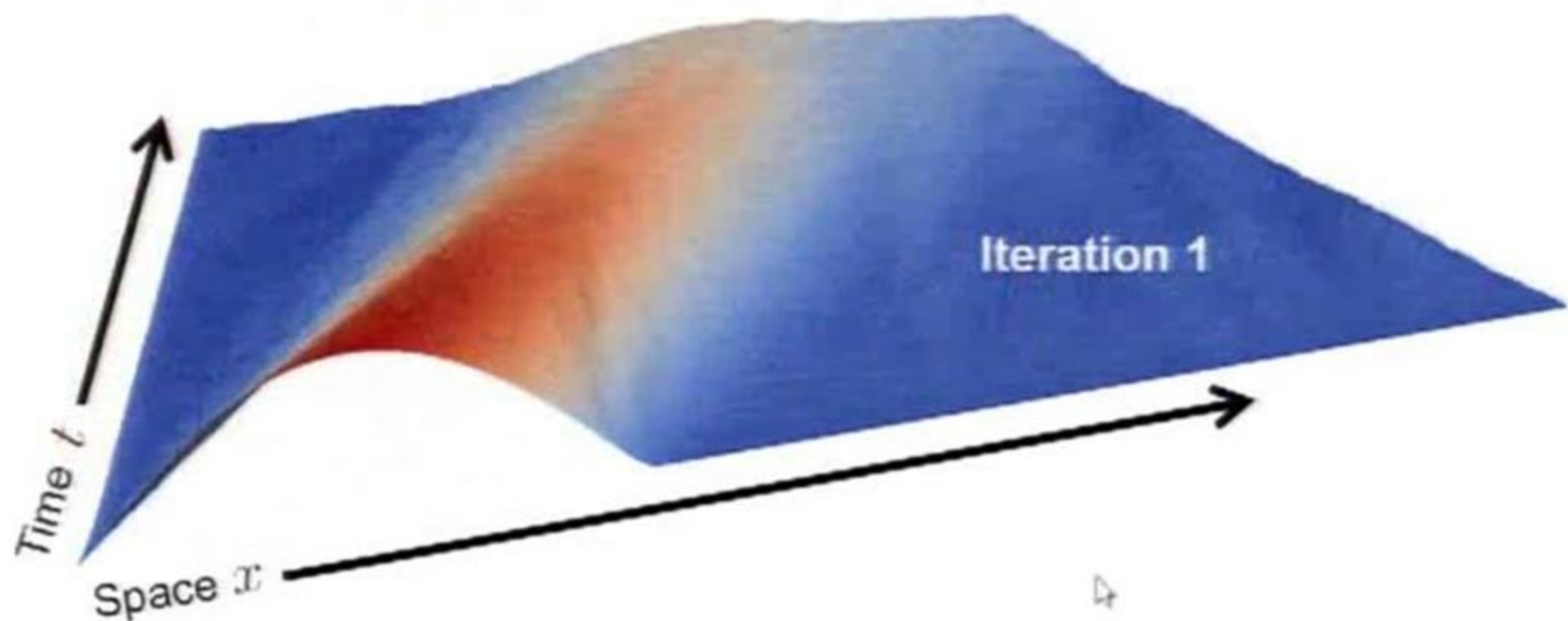
Multigrid-in-time converges to the serial space-time solution in parallel

- Simple advection equation, $u_t = -cu_x$
- Initial condition is a wave



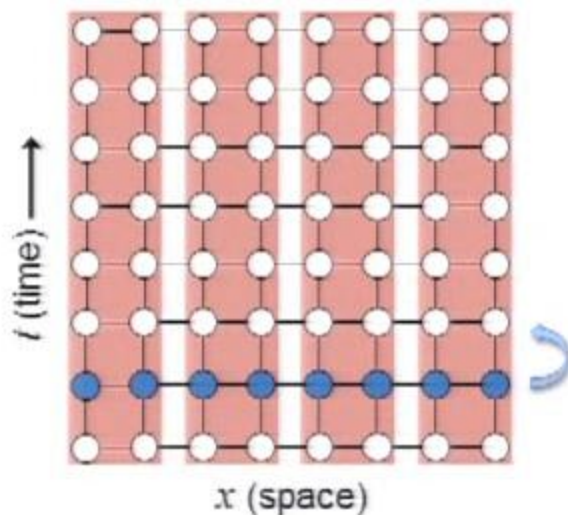
Multigrid-in-time converges to the serial space-time solution in parallel

- Simple advection equation, $u_t = -cu_x$
- Multilevel structure allows for fast data propagation

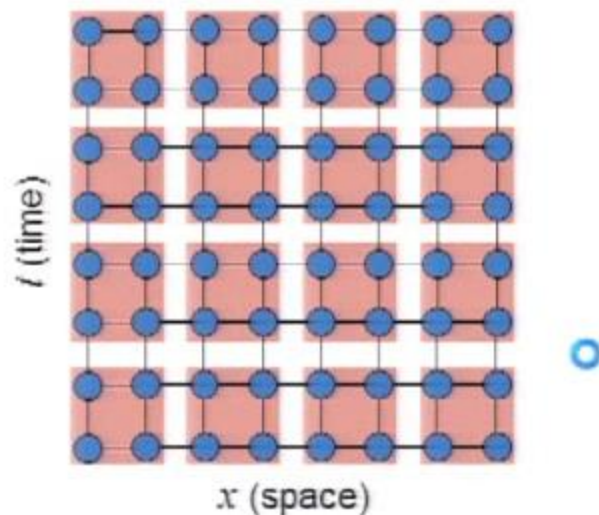


Significantly more parallel resources can be exploited with multigrid in time

Serial time stepping



Multigrid in time



- ➖ Parallelize in **space only**
- ➕ Store **only one time step**

- ➕ Parallelize in **space and time**
- ➖ Store **several time steps**

It's useful to view the time integration problem as a large block matrix system

- General one-step method

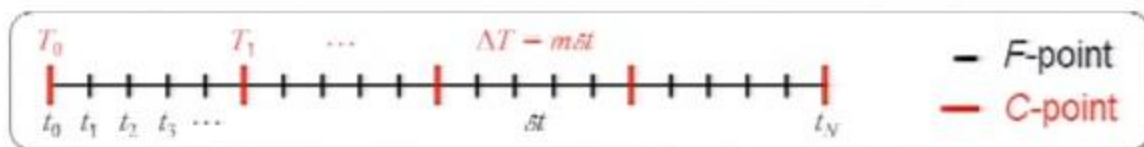
$$\mathbf{u}_i = \Phi_i(\mathbf{u}_{i-1}) + \mathbf{g}_i, \quad i = 1, 2, \dots, N$$

- Linear setting: time marching = block forward solve
 - $O(N)$ direct method, but **sequential**

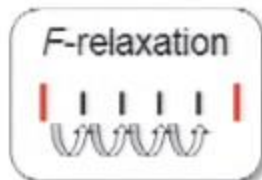
$$A\mathbf{u} \equiv \begin{pmatrix} I & & & & \\ -\Phi & I & & & \\ & \ddots & \ddots & & \\ & & & -\Phi & I \end{pmatrix} \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_N \end{pmatrix} = \begin{pmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_N \end{pmatrix} \equiv \mathbf{g}$$

- Our approach is based on multigrid reduction (MGR) methods (approximate cyclic reduction)
 - $O(N)$ iterative method, but **highly parallel**

MGR dates to 1979 (Ries & Trottenberg) and we are extending it “in time” (MGRIT)



- Relaxation alternates between *F* and *C*-points
 - F-relaxation = integration over each coarse interval
- Coarse system is a time re-discretization
 - Replaces the exact Petrov-Galerkin system
- Non-intrusive approach
 - Time discretization is unchanged
 - User only provides time integrator Φ



Coarse Petrov-Galerkin system is not practical
→ approximate it

$$A_{\Delta} u_{\Delta} = g_{\Delta} = R_{\Phi} g$$

$$A_{\Delta} = \begin{pmatrix} I & & & & \\ -\Phi^m & I & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -\Phi^m & I \end{pmatrix}$$

MGRIT is $O(N)$ for simple parabolic problems, implicit & explicit, 1D-3D

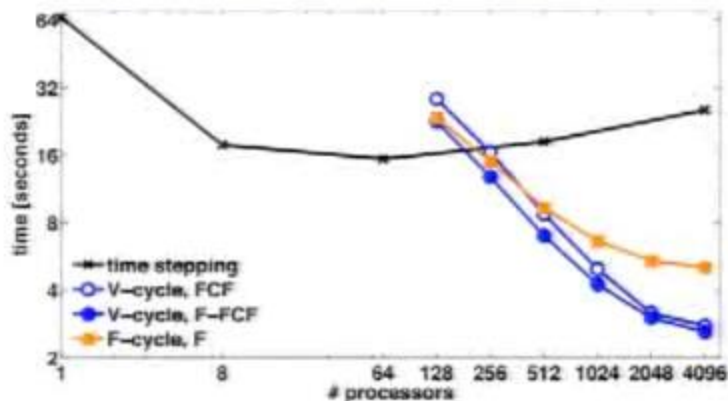
- 2D heat equation, **implicit** backward Euler, F/FCF-relaxation

| Grid = $2^x \times 2^y \times 2^z$ | (4,4,5) | (5,5,7) | (6,6,9) | (7,7,11) | (8,8,13) |
|------------------------------------|---------|---------|---------|----------|----------|
| V-cycles | 10 | 11 | 11 | 11 | 11 |

- Explicit** method: Use either an implicit scheme or spatial coarsening on coarse grids for stability
- Extends to **nonlinear** problems with **FAS** formulation
 - Also $O(N)$ for nonlinear diffusion
- Similar convergence for **any coarsening factor**
 - But larger factors require larger (sequential) F-relaxation intervals
- Two-level** MGRIT with **F**-relaxation is **equivalent to parareal**
 - Popular method, typically not viewed as multigrid

Parallel speedups can be huge, but not in the way we are accustomed

- Parallel time integration is driven entirely by hardware
 - Time stepping is already $O(N)$
- Useful only beyond some parallel scale
 - There is a **crossover point**
 - Need 10-100x more parallelism just to break even
- The **more time steps**, the **more speedup** potential
 - Applications that require lots of time steps will benefit first
 - Speedups (so far) **up to 49x on 100K cores**



3D Heat Equation: $33^3 \times 4097$,
8 procs in space, **6x speedup**

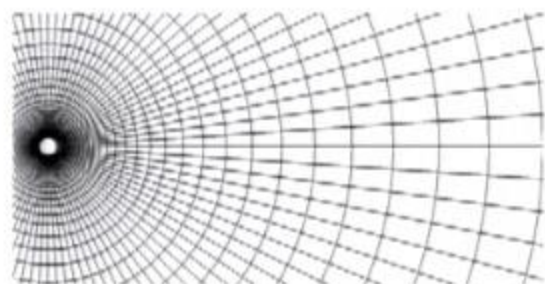
XBraid is open source and designed to be both non-intrusive and flexible

- User defines two objects:
 - *App* and *Vector*
- User also writes several wrapper routines:
 - *Phi*, *Init*, *Clone*, *Sum*, *SpatialNorm*, *Access*, *BufPack*, *BufUnpack*
 - *Coarsen*, *Refine* (optional, for spatial coarsening)
- Example: *Phi(app, u, status)*
 - Advances vector *u* from time *tstart* to *tstop* and returns a target refinement factor
- Code stores only C-points to minimize storage
 - Ability to coarsen by large factors means fewer parallel resources
 - Memory multiplier per processor $\sim O(\log N)$ with time coarsening, $O(1)$ with space-time coarsening

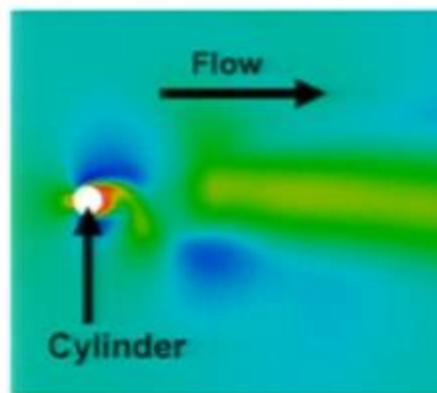


Compressible Navier-Stokes (nonlinear) – speedups to 7.5x with typical MG scaling

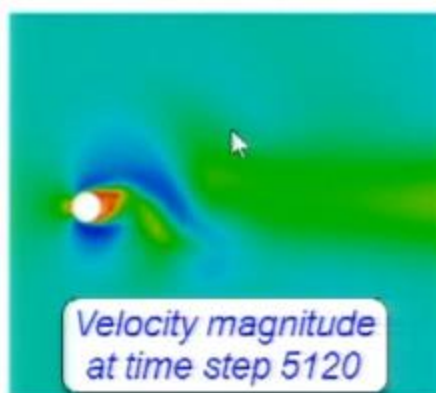
- Coupled Xbraid with existing code Strand2D (DOD project)
 - ~ 500 lines of Xbraid wrapper code plus minor changes to Strand2D
 - ~ 3 weeks with minimal outside help



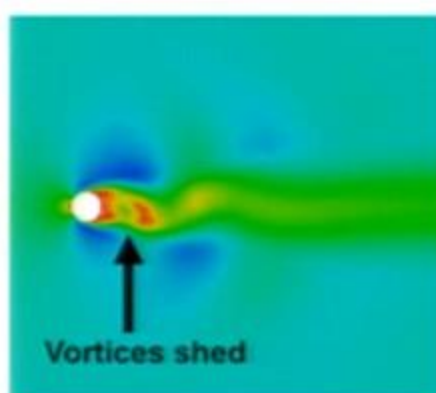
Iteration 1



Iteration 5

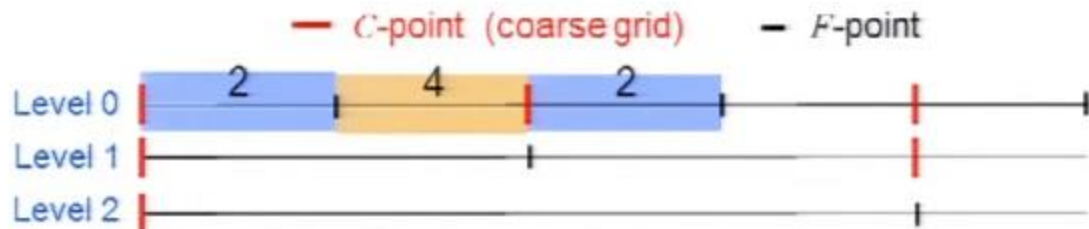


Iteration 13

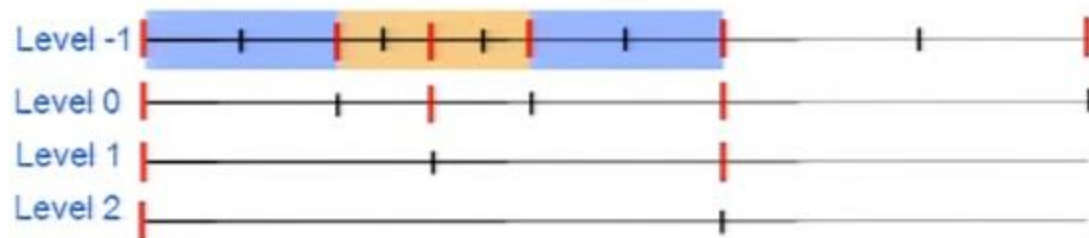


XBraid uses FMG to get adaptivity in time (and space) – not yet fully implemented

- User returns refinement factor in $\Phi(t)$
- Example time grid hierarchy



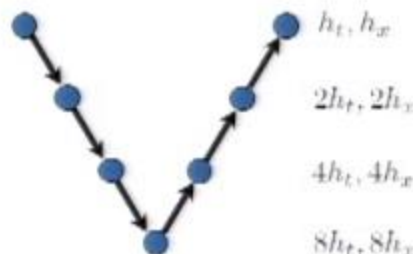
- User requests refinement factors on the finest grid which generates a new grid and hierarchy



Notice
new
C-pts

Space-time coarsening, multi-step methods, and future work

- **Space-time coarsening** speeds up computations and greatly reduces memory use
 - User writes optional *Coarsen()* & *Refine()*
 - Works well for **implicit diffusion**
 - Convergence degrades for **explicit diffusion** near CFL
- **Multi-step** methods
 - Convert to one-step by grouping unknowns
- Other problems and applications
 - **Hyperbolic problems** and CFD (DOD project)
 - Power grid application
- Pursue fairly non-intrusive approaches in XBraid (if possible)
- **Full space-time** multigrid and **Newton**-based approaches
 - More intrusive, but can be more effective



Nearly 50 years of research exists but has only scratched the surface

- **Earliest work** goes back to 1964 by Nievergelt
 - Led to multiple shooting methods, Keller (1968)
- **Space-time multigrid** methods for parabolic problems
 - Hackbusch (1984); Horton (1992); Horton and Vandewalle (1995)
 - The latter is one of the first **optimal & fully parallelizable** methods to date
- **Parareal** was introduced by Lions, Maday, and Turincini in 2001
 - Probably the most widely studied method
 - Gander and Vandewalle (2007) show that parareal is a **two-level FAS multigrid** method
- **Discretization specific** work includes
 - Minion, Williams (2008, 2010) – PFASST, spectral deferred correction / parareal
 - DeSterck, Manteuffel, McCormick, Olson (2004, 2006) – FOSLS
- **Research on these methods is ramping up!**
 - Ruprecht, Krause, Speck, Emmett, Langer, ... **this is not an exhaustive list**

Our Parallel MG Research Team



Veselin
Dobrev

Rob
Falgout

Hormazd
Gahvari

Tzanio
Kolev

Daniel
Osei-Kuffuor

Anders
Pettersson

Jacob
Schroder

Panayot
Vassilevski

Lu
Wang

Ulrike
Yang

■ University collaborators and summer interns

- CU Boulder (Manteuffel, McCormick, Ruge, Jones, O'Neill, Mitchell, Southworth), Penn State (Brannick, Xu, Zikatanov), UCSD (Bank), Ball State (Lvshits), U Wuppertal (Kahl), Memorial University (MacLachlan), U Illinois (Gropp, Olson, Rienz), KU Leuven (Friedhoff), U Cologne (Lanscr)

Some relevant mini-symposiums

- **Multigrid Methods**

- Tuesday 10:00 AM / 2:15 PM (MS202 / MS226): [Advances in Multigrid Methods and Their Applications - Parts I and II](#), Room 259
- Tuesday 4:25 PM (MS254): [Advancements in Generalizing Algebraic Multigrid Methods](#), Room 151G

- **Parallel Time Integration**

- Wednesday 10:55 AM / 2:00 PM (MS261 / MS286): [Parallel Methods for Time Integration - Parts I and II](#), Room 250E

- **There are many other related talks and posters scattered throughout the program, so please look for them**



Summary and Conclusions

- Multigrid methods are ideal for exascale
 - Optimal, naturally resilient to faults, minimize data movement
- Parallel time integration is needed on future architectures
 - Major paradigm shift for computational science!
- Success with approaches for reducing communication
 - Non-Galerkin AMG, Mult-additive AMG, AMG-DD
- MGRIT algorithm extends multigrid reduction “in time”
 - Non-intrusive yet flexible approach (open-source code XBraid)
 - Demonstrated speedups for parabolic problems and CFD
 - Exploring other application areas

Thank You!



U.S. DEPARTMENT OF
ENERGY

Office of
Science