SciDAC
Scientific Discovery through Advanced Computing

Multigrid at Extreme scales: Communication Reducing Data Models and Asynchronous Algorithms

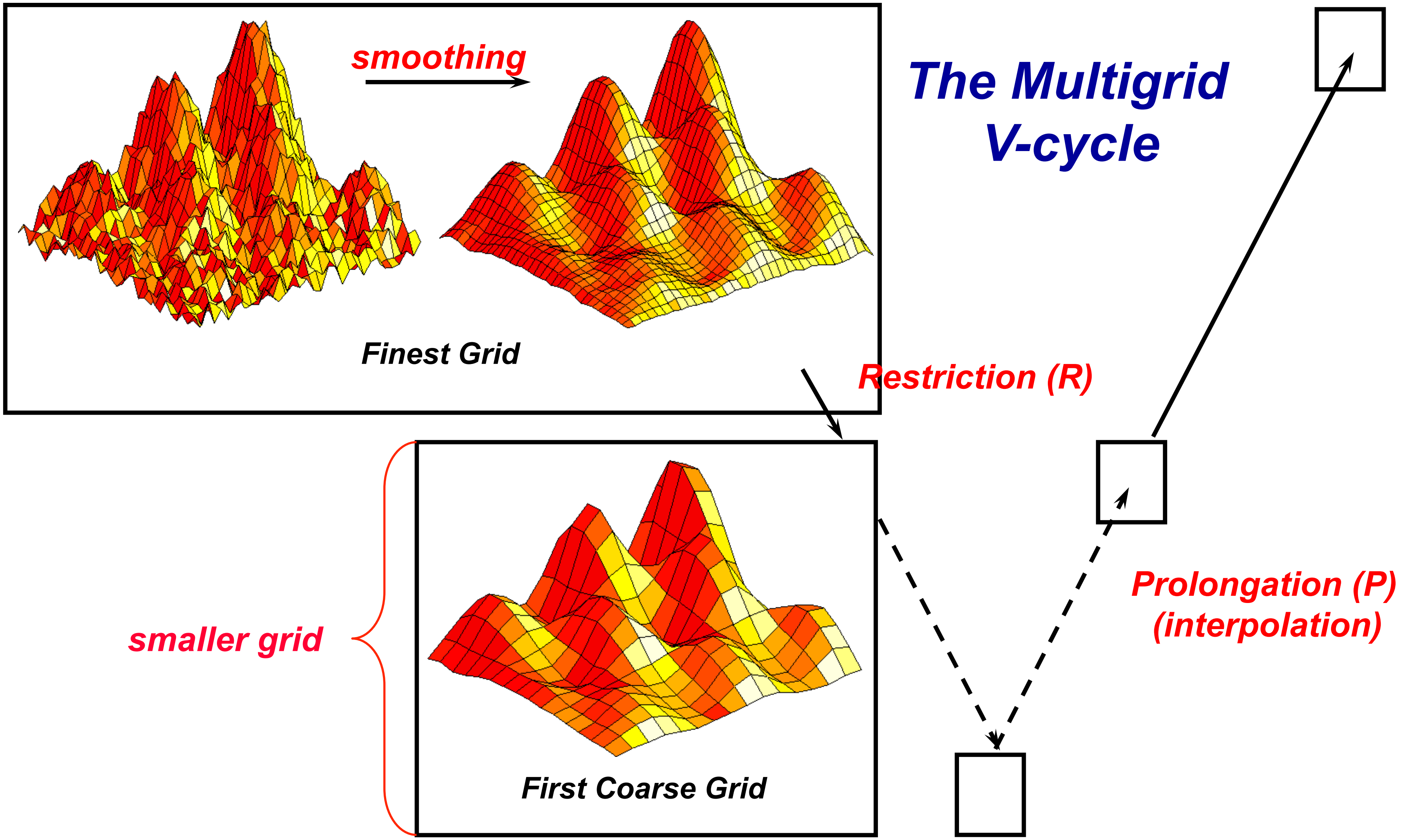
Mark Adams

Columbia University

Outline

- Establish a lower bound on solver complexity
  - Apply ideas to Magnetohydrodynamics (MHD)
- Distributed memory & communication avoiding MG
  - Asynchronous unstructured Gauss-Seidel
- New algebraic multigrid (AMG) in PETSc
  - Application to 3D elasticity and 2D Poisson solves
- Data centric MG: cache aware & communication avoiding
  - Application to 2D 5-point stencil V(1,1) cycle

Multigrid motivation: smoothing and coarse grid correction



smoothing

Finest Grid

The Multigrid V-cycle

Restriction (R)

smaller grid

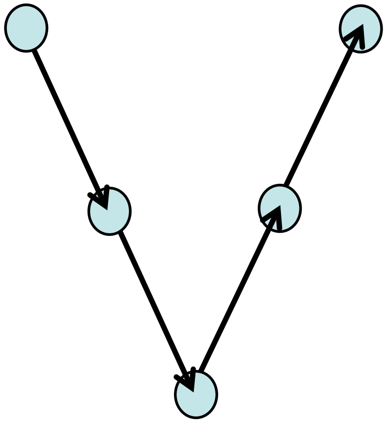
First Coarse Grid

Prolongation (P) (interpolation)

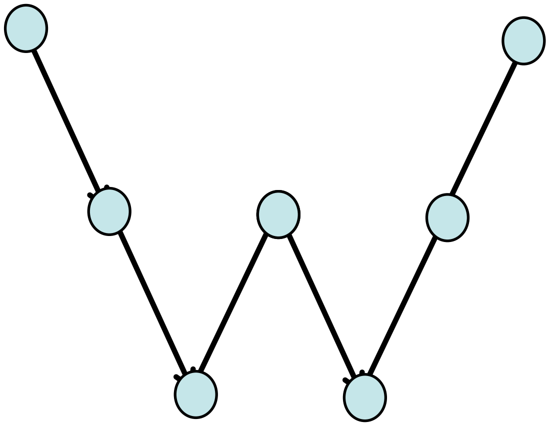
Option:UCRL#

SciDAC
Scientific Discovery through Advanced Computing

Multigrid Cycles

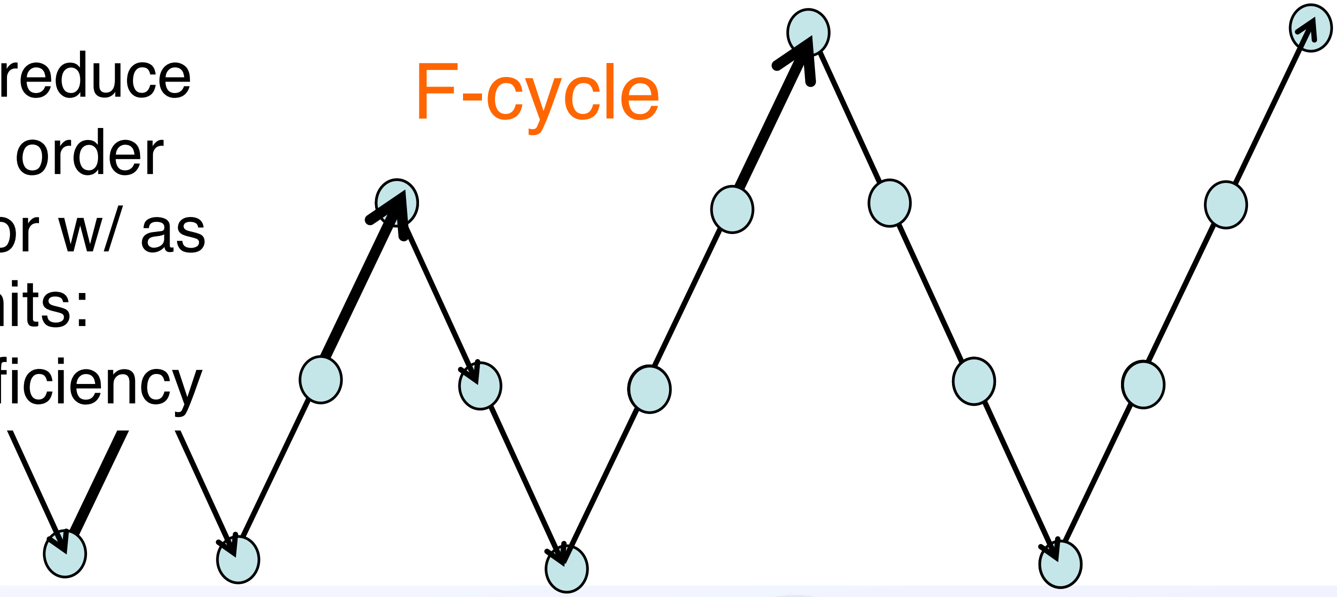


V-cycle



W-cycle

F-cycle



One F-cycle can reduce algebraic error to order discretization error w/ as little as 5 work units: “textbook” MG efficiency

Discretization error in one F-cycle (Bank, Dupont, 1981)

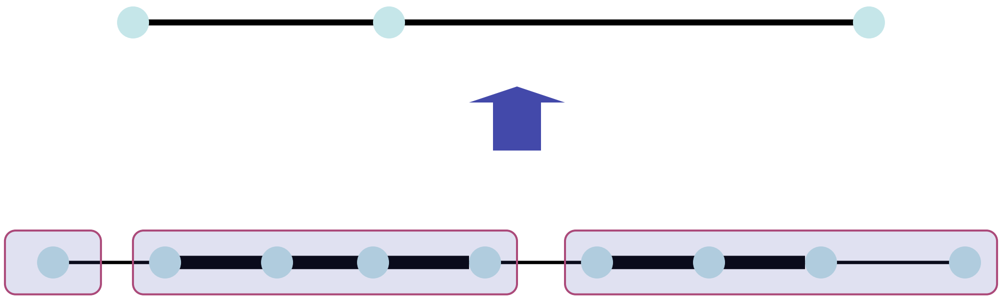
- Define error: $E(x) \le E_d(x) + E_a(x)$ (discrete. + algebraic)
- Assume error $E_d(x) \le Ch^p$ (point-wise theory)
- Example: 2nd ($p=2$) order discretization & coarsening factor of 2.
- Induction hypothesis: require $r \ge E_a/E_d$ (eg, $r=½$)
- Define Γ rate *error* reduction of solver (eg, 0.1 w/ a V-cycle)
  - Can *prove* this or *determine experimentally*
  - No Γ w/defect correction – can use Γ of low order method.
- Use induction: Error from coarse grid: $C(2h)^2 + r \bullet C(2h)^2$
  - Alg. Err. Before V-cycle: $E_a < C(2h)^2 + r \bullet C(2h)^2 - Ch^2$
    - Actually should be $+Ch^2$ but sign of error should be same
  - And we want $\Gamma \bullet E_a = \Gamma \bullet (C(2h)^2 + r \bullet C(2h)^2 - Ch^2) < r \bullet E_d \le r \bullet Ch^2$
  - $\Gamma = r/(4r + 3)$, 1 equation, 2 unknowns … fix one:
    - eg, $r = ½ \Rightarrow \Gamma = 0.1$
    - If you want to use $+ Ch^2$ term then its $\Gamma = r/(4r + 5)$

Multigrid V(ν_1,ν_2) & F(ν_1,ν_2) cycle

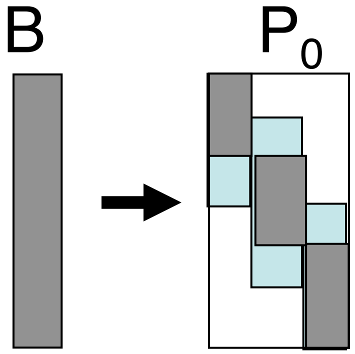
```
▪ function u = MGV(A,f)
    • If A coarsest grid
        – $u \leftarrow A^{-1}f$
    • else
        – $u \leftarrow S^{\nu 1}(f, 0)$                  -- Smoother (pre)
        – $r_H \leftarrow P^T( f – Au )$
        – $e_H \leftarrow MGV( P^TAP, r_H )$          -- recursion (Galerkin)
        – $u \leftarrow u + Pe_H$
        – $u \leftarrow S^{\nu 2}(f, u)$                  -- Smoother (post)

▪ function u = MGF($A_i$,f)
    • if $A_i$ is coarsest grid
        – $u \leftarrow A_i^{-1}f$
    • else
        – $r_H \leftarrow R\ f$
        – $e_H \leftarrow FGV( A_{i-1}, r_H )$          -- recursion
        – $u \leftarrow \mathbf{P}e_H$
        – $u \leftarrow u + MGV( A_i, f – A_iu )$
```

Algebraic multigrid (AMG) - Smoothed Aggregation

- MG requires a smoother and coarse grid space
  - Columns of P
- Piecewise constant functions are easy
  - "Plain" aggregation
- Nodal aggregation, or partitioning
- Example: 1D 3-point stencil



Kernel vectors of operator (B)



"Smoothed" aggregation: lower energy of functions

For example: one Jacobi iteration: $P \leftarrow (I - \omega D^{-1} A) P_0$

Outline

- Establish a lower bound on solver complexity
  - Apply ideas to Magnetohydrodynamics (MHD)
- Distributed memory & communication avoiding MG
  - Asynchronous unstructured Gauss-Seidel
- New algebraic multigrid (AMG) in PETSc
  - Application to 3D elasticity and 2D Poisson solves
- Data centric MG: cache aware & communication avoiding
  - Application to 2D 5-point stencil V(1,1) cycle

Compressible resistive MHD equations in strong conservation form

$$\frac{\partial U}{\partial t} + \boxed{\frac{\partial F_j(U)}{\partial x_j}} = \boxed{\frac{\partial \tilde{F}_j(U)}{\partial x_j}}$$

Diffusive

Hyperbolic

$$\tau_{ij} = \rho\nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}\right)$$

$$U = \{\rho, \rho u_i, B_i, e\}^T$$

$$e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u_i u_i + \frac{1}{2}B_i B_i$$

$$F_j(U) = \left\{\begin{array}{c} \rho u_j \\ \rho u_i u_j + p\delta_{ij} + \frac{1}{2}B_k B_k \delta_{ij} - B_i B_j \\ u_j B_i - B_j u_i \\ \left(e + p + \frac{1}{2}B_k B_k\right) u_j - B_i u_i B_j \end{array}\right\}$$

$$\tilde{F}_j(U) = \left\{\begin{array}{c} 0 \\ Re^{-1}\tau_{ij} \\ S^{-1}\eta\left(\frac{\partial B_i}{\partial x_j} + \frac{\partial B_j}{\partial x_i}\right) \\ S^{-1}\eta\left(\frac{1}{2}\frac{\partial B_i B_i}{\partial x_j} - B_i\frac{\partial B_j}{\partial x_i}\right) + Re^{-1}\tau_{ij}u_i + Pe^{-1}\kappa\frac{\partial T}{\partial x_j} \end{array}\right\}$$

Reynolds no.

Lundquist no.

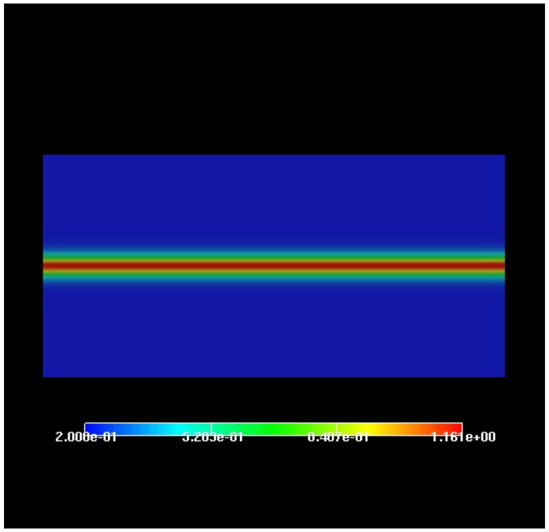
Peclet no.

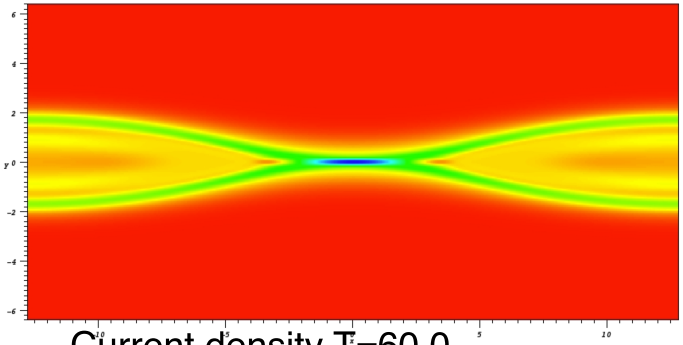
Fully implicit resistive compressible MHD Multigrid – back to the 70’s

- Geometric MG, Cartesian grids
  - Piecewise constant restriction R, linear interpolation (P)
- Red/black point Gauss-Seidel smoothers
  - Requires inner G-S solver be coded
- F-cycle
  - Two V(1,1) cycles at each level
  - Algebraic error < discretization error in one F-cycle iteration
- Matrix free - more flops less memory
  - Memory increasingly bottleneck - Matrix free is way to go
  - processors (cores) are cheap
    - memory architecture is expensive and slow (relative to CPU)
- Non-linear multigrid
  - No linearization required
- Defect correction for high order (L_2) methods
  - Use low order discretization (L_1) in multigrid solver (stable)
  - Solve $L_1 x^{k+1} = f - L_2 x^k + L_1 x^k$

Magnetic reconnection problem

- GEM reconnection test
  - 2D Rectangular domain, Harris sheet equilibrium
  - Density field along axis: (fig top)
  - Magnetic (smooth) step
  - Perturb B with a “pinch”
- Low order preconditioner
  - Upwind - Rusanov method
- Higher order in space: C.D.
- Solver
  - 1 F-cycle w/ 2 x V(1,1) cycles per time step
    - Nominal cost of 9 explicit time steps
    - ~18 work units per time step
- Viscosity:
  - Low: μ=5.0D-04, η=5.0D-03, κ=2.0D-02
  - High: μ=5.0D-02, η=5.0D-03, κ=2.0D-02
- B_z: B_z=0 and B_z=5.0
  - Strong guide field B_z (eg, 5.0)
  - critical for tokomak plasmas

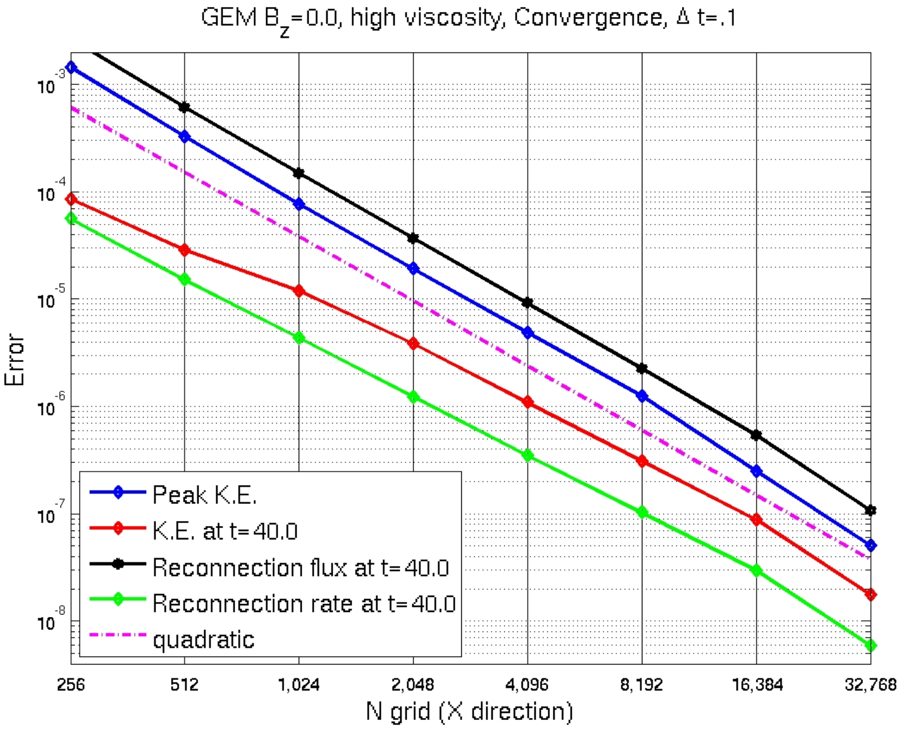


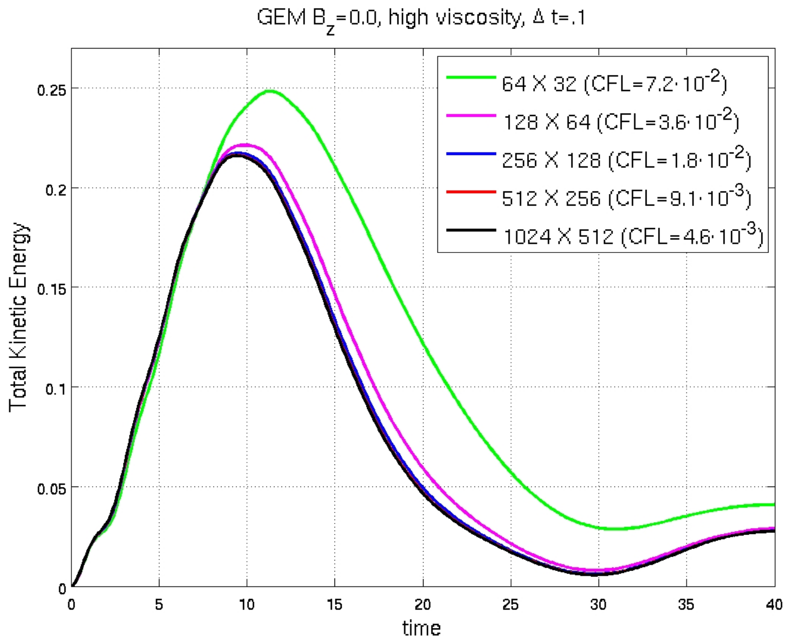


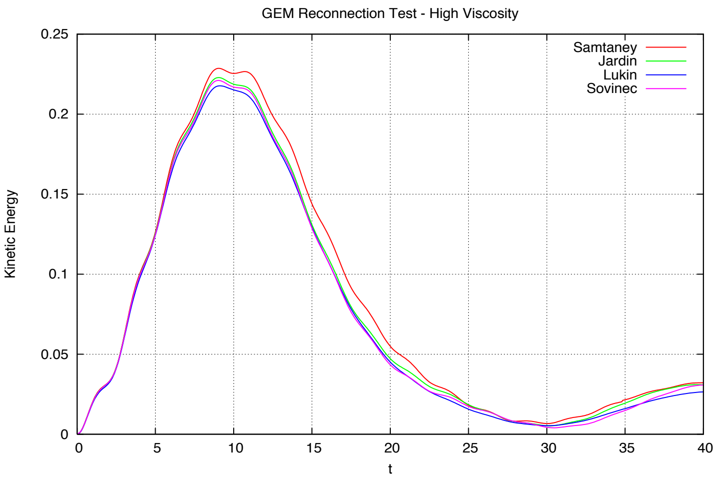
Current density T=60.0

B_z = 0, High viscosity

- Time = 40.0, Δt = 1.
  - ~100x CFL on 512 X 256 grid
- 2nd order spatial convergence
- Backward Euler in time
- Benchmarked w/ other codes
- Convergence studies (8B eqs)



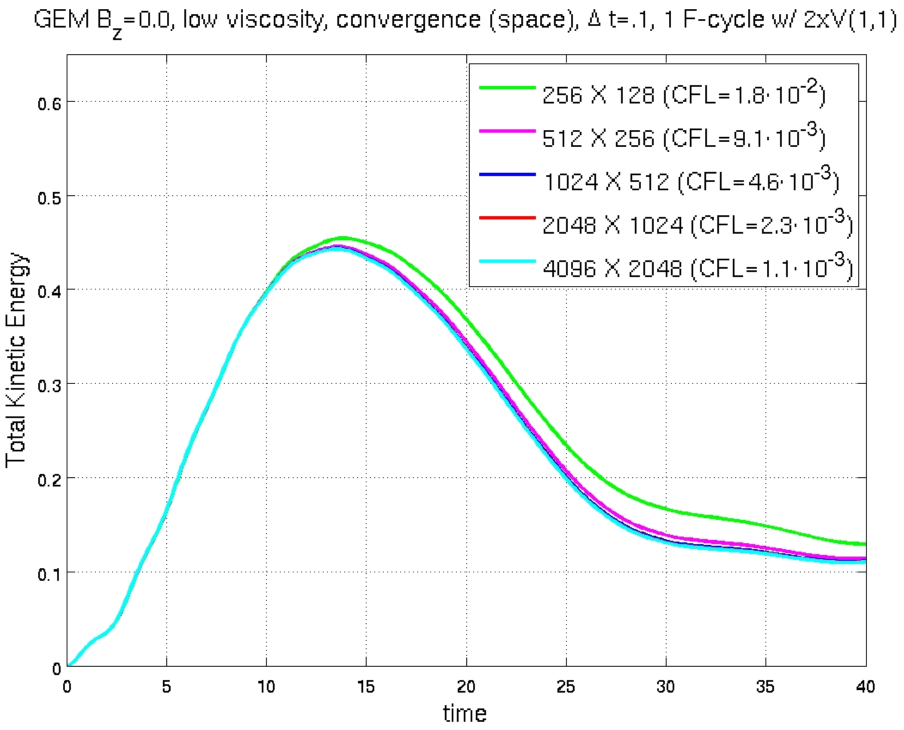


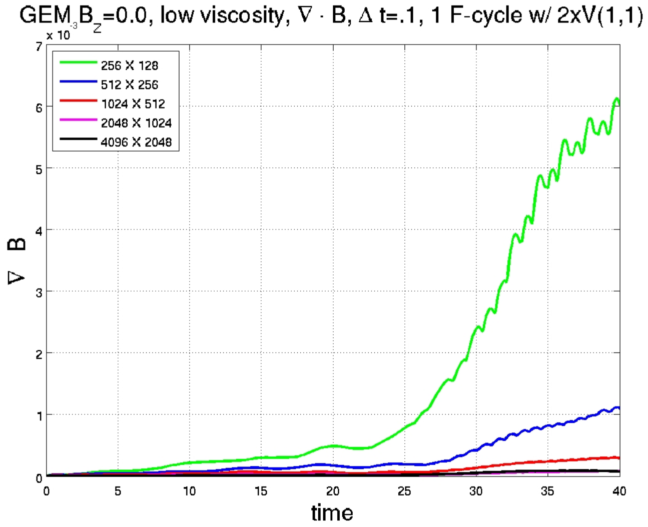


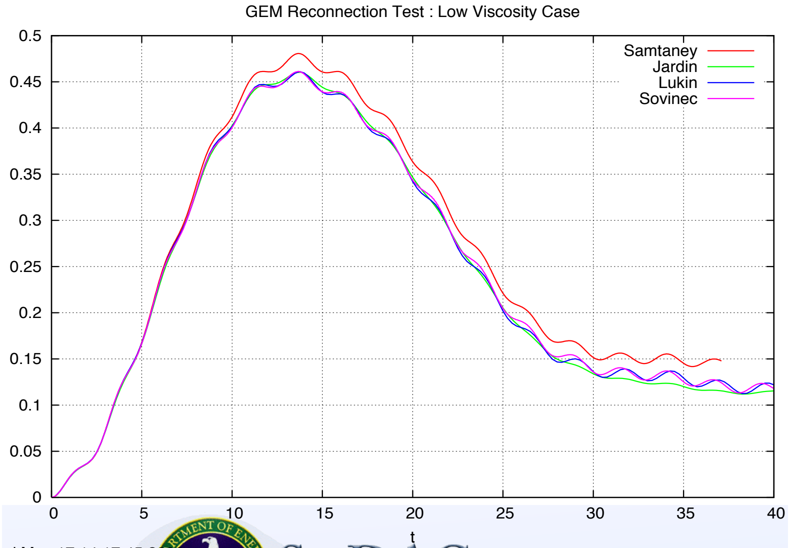
Tue May 09 07:57:02 2006

$B_z = 0$, Low viscosity, $\nabla \cdot B = 0$

- Time = 40.0, $\Delta t = .1$
- 2nd order spatial convergence
- $\nabla \cdot B = 0$ converges
- Kin. E compares well w/ other codes



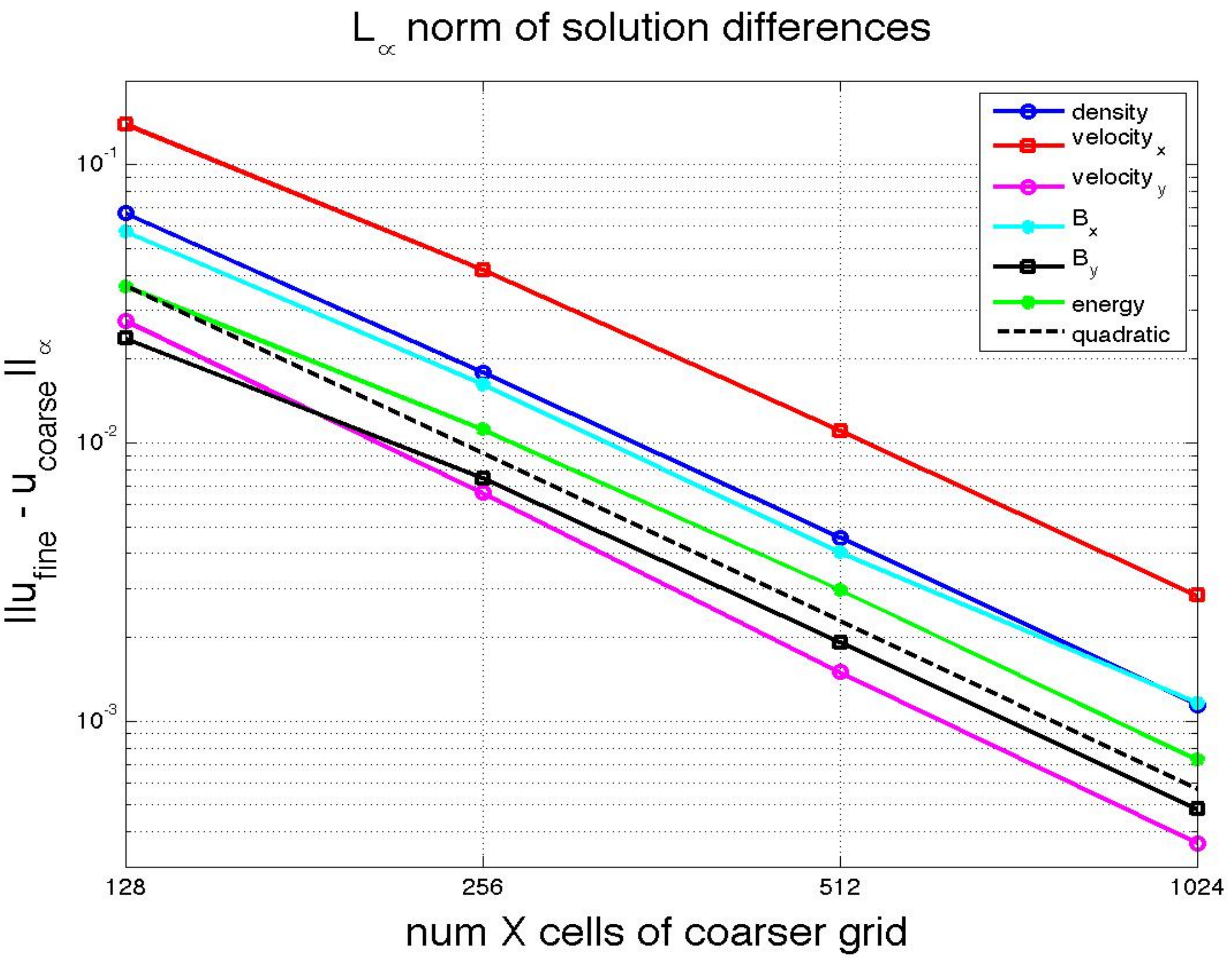




ed May 17 14:17:45 20

Solution Convergence

μ=1.0D-03, η=1.0D-03, B_z=0

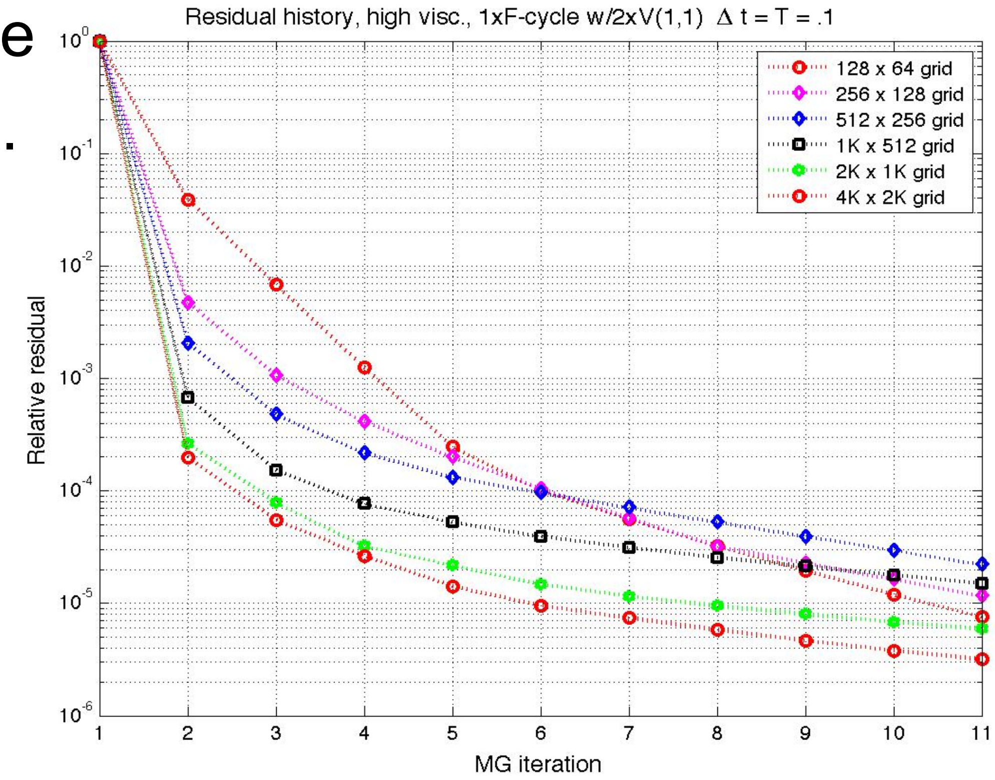


Option:UCRL#

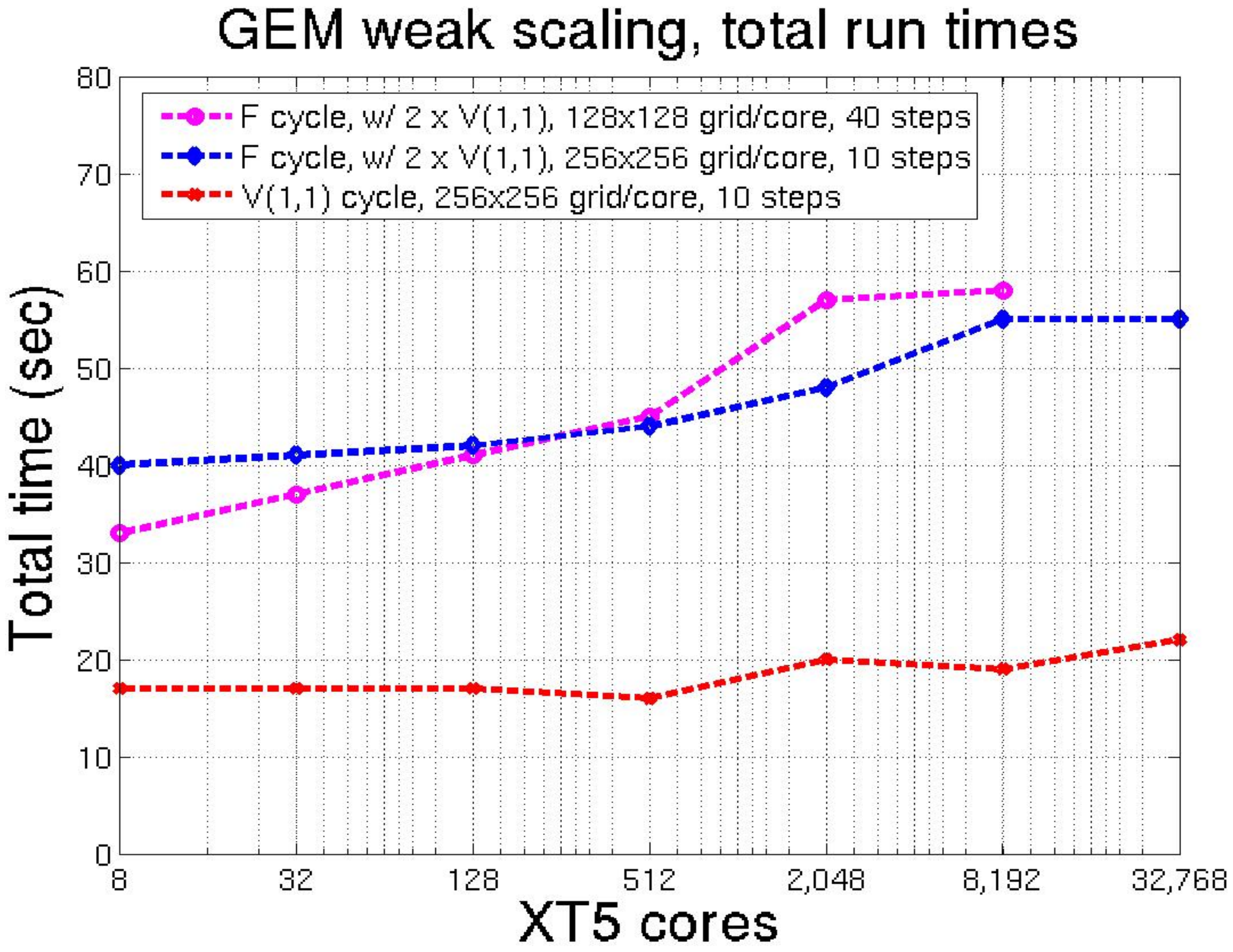
SciDAC
Scientific Discovery through Advanced Computing

Residual history

- Residual history (1st time step), high viscosity, B = 0
- F cycles achieve discretization error
  - Super convergence
- No Γ w/defect correct.
- Use Γ for L_1



Weak scaling – Cray XT-5



Option:UCRL#

SciDAC
Scientific Discovery through Advanced Computing

Outline

- Establish a lower bound on solver complexity
  - Apply ideas to Magnetohydrodynamics (MHD)
- Distributed memory & communication avoiding MG
  - Asynchronous unstructured Gauss-Seidel
- New algebraic multigrid (AMG) in PETSc
  - Application to 3D elasticity and 2D Poisson solves
- Data centric MG: cache aware & communication avoiding
  - Application to 2D 5-point stencil V(1,1) cycle

What do we need to make multigrid fast & scalable at exa-scale?

- Architectural assumptions:
  - Distributed memory message passing is here for a while
  - Future growth will be primarily on the “node”
  - Memory bandwidth to chip can not keep up with processing speed
    - Need higher computational intensity - “flops are free”…
- Multigrid issues:
  - Distributed memory network (latency) is still critical (if not hip)
    - Growth is on the node but distributed memory dictates data structures, etc.
      - Node optimizations can be made obsolete after distributed data structures added
    - Applications must use good distributed data models and algorithms
    - *Coarse grids must me partitioned carefully - especially with F-cycles*
      - *Coarse grids put most pressure on network*
    - Communication avoiding algorithms are useful here
      - But tedious to implement – *need support compliers, source–to-source, DSLs, etc.*
  - Computational intensity is low - increase with loop fusion (or streaming HW?)
    - Textbook V(1,1) multigrid does as few as 3 work unites per solve
      - Plus a restriction and interpolation.
    - Can *fuse* one set of 2 (+restrict.) & one set of 1 (+ interp.) of these loops
    - Communication avoiding can be added … *data centric multigrid*

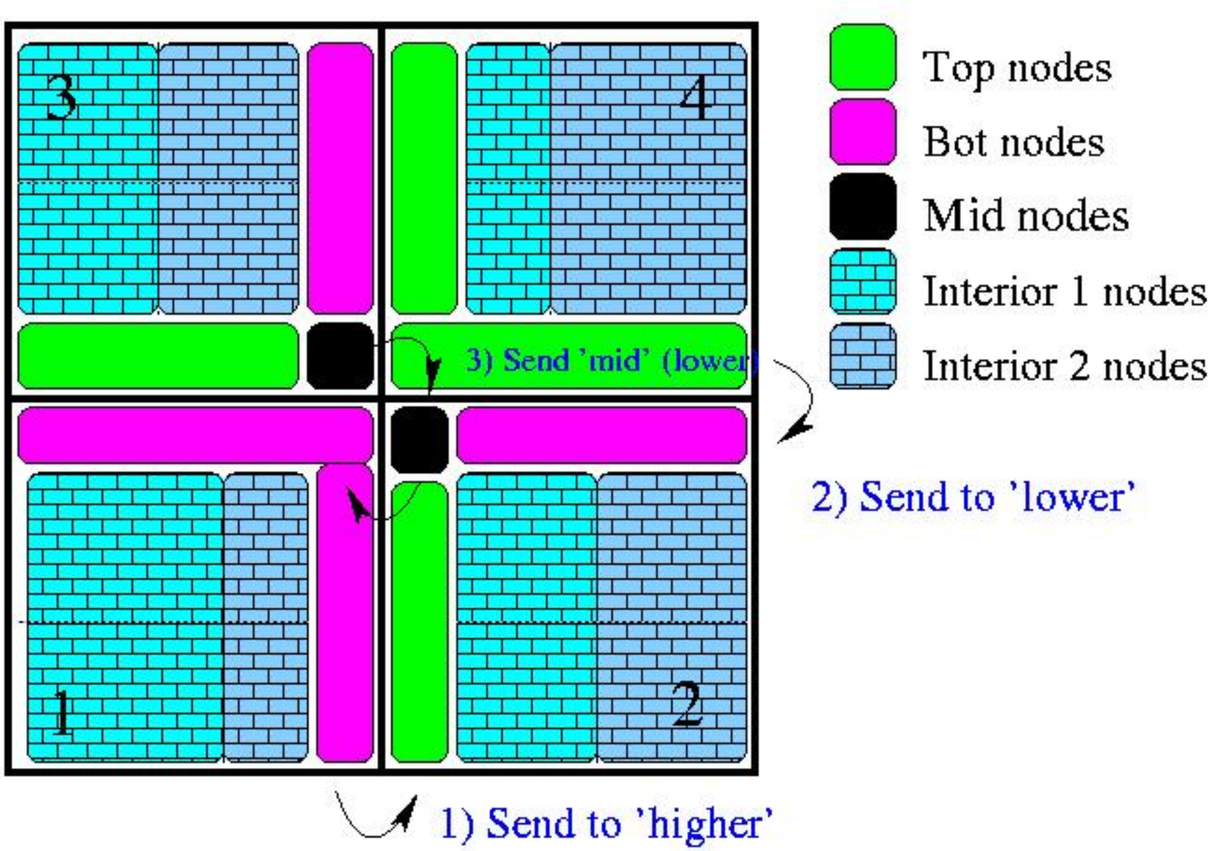
Outline

- Establish a lower bound on solver complexity
  - Apply ideas to Magnetohydrodynamics (MHD)
- Distributed memory & communication avoiding MG
  - Asynchronous unstructured Gauss-Seidel
- New algebraic multigrid (AMG) in PETSc
  - Application to 3D elasticity and 2D Poisson solves
- Data centric MG: cache aware & communication avoiding
  - Application to 2D 5-point stencil V(1,1) cycle

Case study: Parallel Gauss-Seidel Algorithm

- Standard CS algorithm (bulk synchronous) graph coloring:
  - Color graph and for each color:
    - Gauss-Seidel process vertices
    - communicate ghost values (soft synchronization)
- 3, 5, 7 point stencil (1D, 2D, 3D) just two colors (not bad)
- 3D hexahedra mesh: 13+ colors (lots of synchronization)
  - General coloring also has pathological cache behavior
- Exploit domain decomposition + nearest neighbor graph property (data locality) + static partitioning
- Instead of computational depth 13+
  - have computational depth about 4+ (3D)
    - The number of processors that a vertex talks to
      - Corners of tiling
- Completely asynchronous algorithm

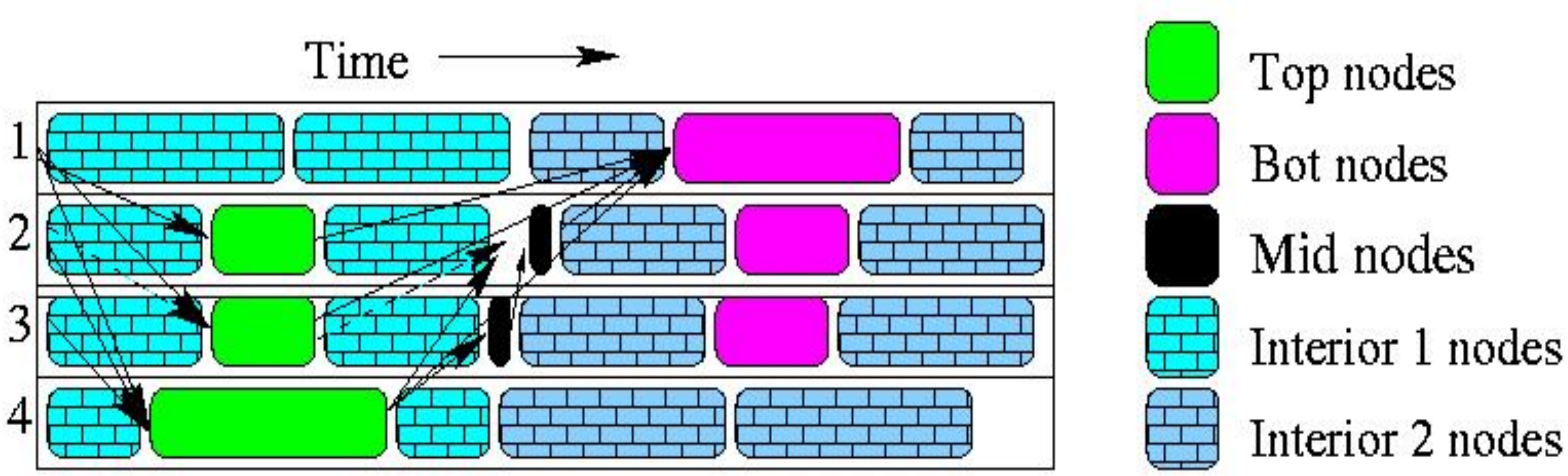
Locally Partition (classify) Nodes



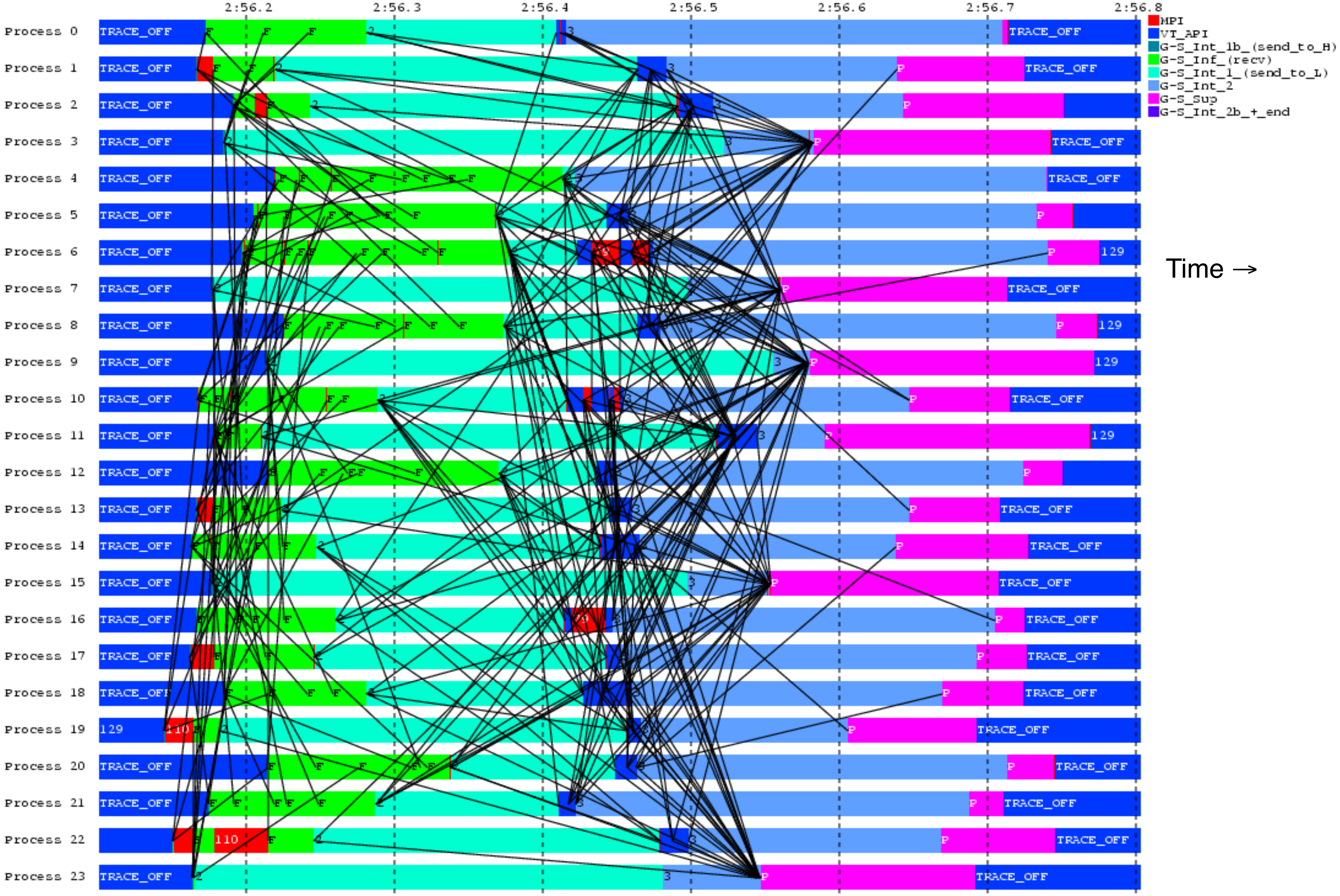
} Boundary nodes

} Interior nodes

Schematic Time Line
Note: reversible

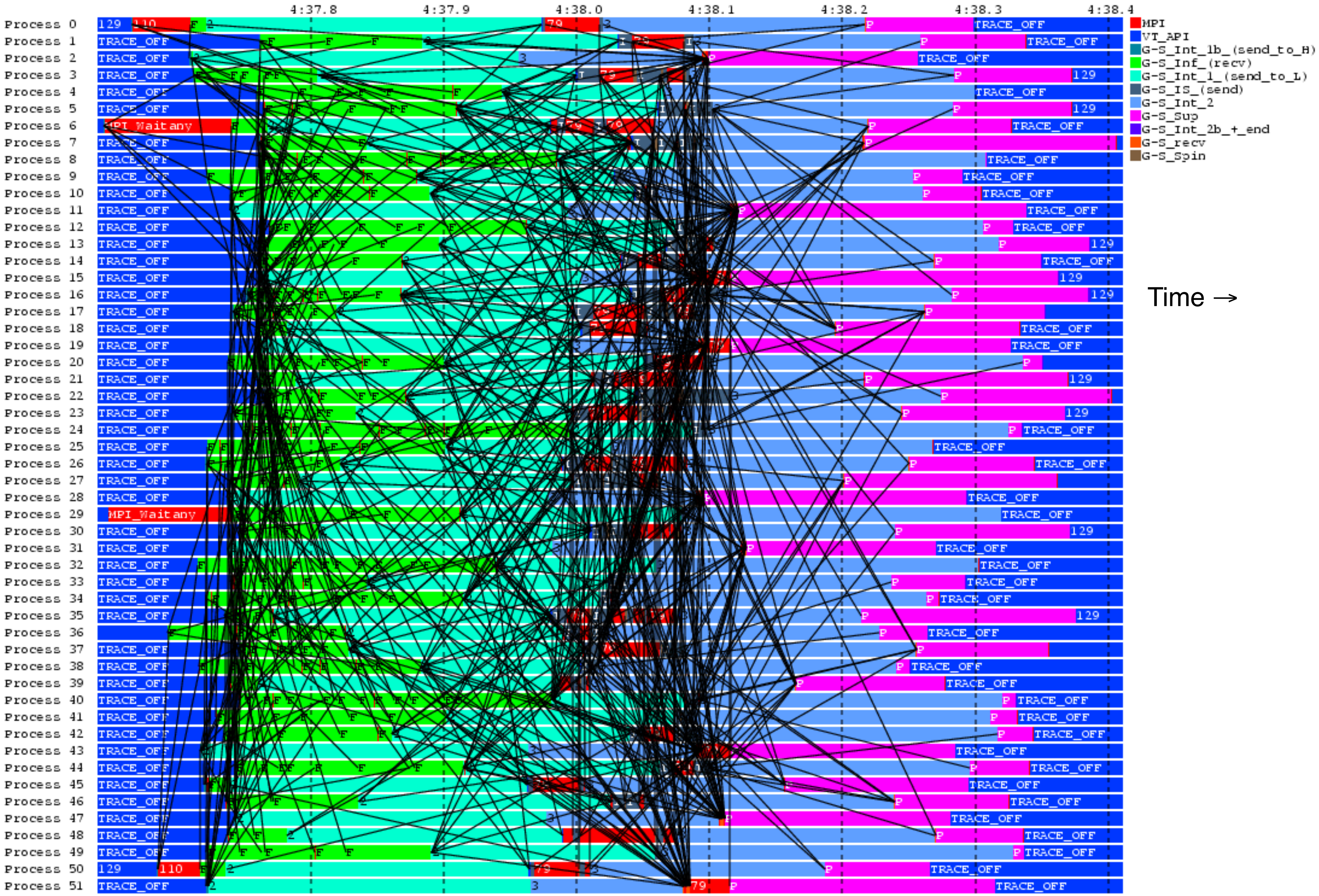


Cray T3E - 24 Processors – About 30,000 dof Per Processor



Time →

Cray T3E - 52 processors – about 10,000 nodes per processor



Lesson to be learned form parallel G-S

- Exploit finite sized domains
  - Domains of order stencil width
- Exploit static partitioning to coordinate parallel processing
- Technique applicable to any level of memory hierarchy
- Overlap communication and computation
- Exploit “surface to volume” character of PDE graphs

Outline

- Establish a lower bound on solver complexity
  - Apply ideas to Magnetohydrodynamics (MHD)
- Distributed memory & communication avoiding MG
  - Asynchronous unstructured Gauss-Seidel
- New algebraic multigrid (AMG) in PETSc
  - Application to 3D elasticity and 2D Poisson solves
- Data centric MG: cache aware & communication avoiding
  - Application to 2D 5-point stencil V(1,1) cycle

Implementations

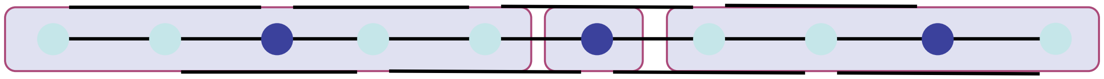
- These ideas implemented in parallel FE framework Olympus & AMG solver Prometheus
  - Gordon Bell prize 2004.
- And in new unstructured geometric MG & smoothed aggregation AMG implementation in PETSc (PC GAMG):
  - -pc_type gamg –pc_gamg_type sa
  - Rely on common parallel primitives to
    - Reduce code size
    - Amortize cost of optimization & of porting to new architectures/PMs
  - PETSc has rich set of common parallel primitives:
    - GAMG ~2,000 lines of code
    - Prometheus ~25,000 lines of code
      - About 20K of this implements GAMG functionality

New aggregation algorithm for SA

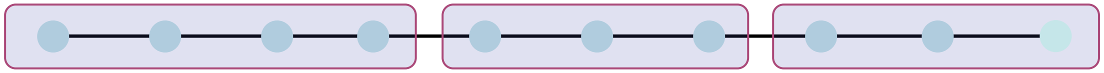
- My old aggregation algorithm is complex, don’t want to reimplement, want to use standard PETSc primitives if possible
- Independent sets are useful in coarsening
  - Independent set: set of vertices w/o edges between each other
  - Maximal: can not add a vertex and still be independent
- MIS(k) (MIS on A^k) algorithm is well defined & good parallel algorithms
  - “Greedy” MIS algorithms naturally create aggregates
- Rate of coarsening critical for complexity
  - Slow coarsening helps convergence at expense of coarse grid complxty
  - Optimal rate of coarsening for SA for 2nd order FEM is 3x
    - Recovers geometric MG in regular grid
    - Results in no stencil growth on regular grids
- MIS(2) provides a decent coarsening rate for unstructured grids
- MIS/greedy aggregation can lead to non-uniform aggregate sizes
- New “aggregation smoothing” with precise parallel semantics and use of MIS primitives.

New aggregation algorithm for SA

- Drop small edges from graph G induced by matrix
  - $G = D^{-½}(AA^T)D^{-½}$
  - If $G_{ij} < \theta$, then drop from Graph (eg, $\theta = 0.05$)
- Use MIS(2) on G to get initial aggregates



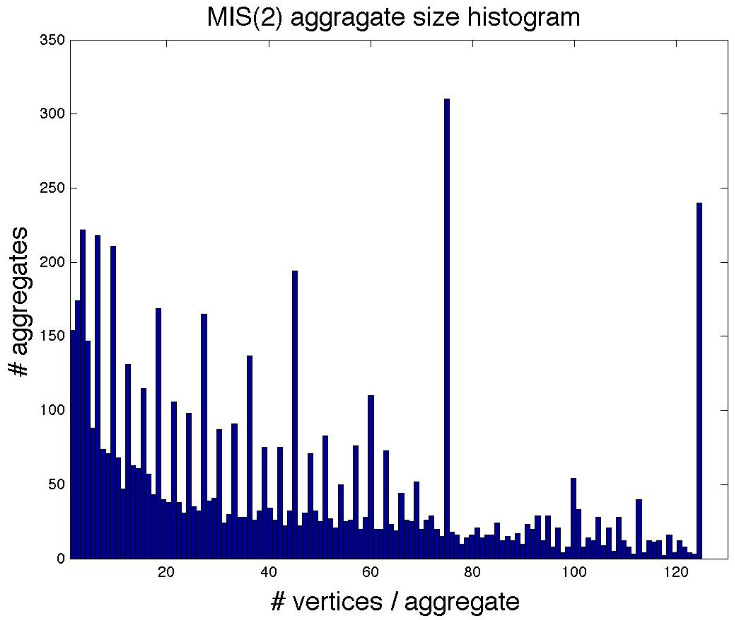
- Greedy (MIS(1) like algorithm) modified aggregates

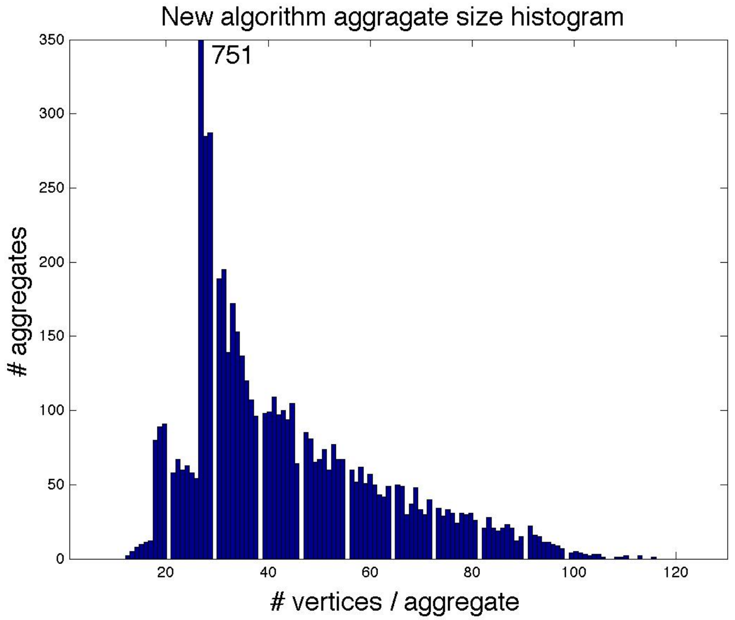


Results of new algorithm Histogram of aggregate sizes

64^3 Mesh (262144 nodes)

First order hex mesh of cube





Weak Scaling of SA on 3D elasticity

Cray XE-6 (Hopper)

- Weak scaling of cube
  - 81,000 eqs / core
- 8 node “brick” elements
- F-cycles
- Smoothed aggregation
- 1 Chebyshev pre & post smoothing
- Dirichlet on one face only
- Uniform body force parallel to Dirichlet plane

Performance

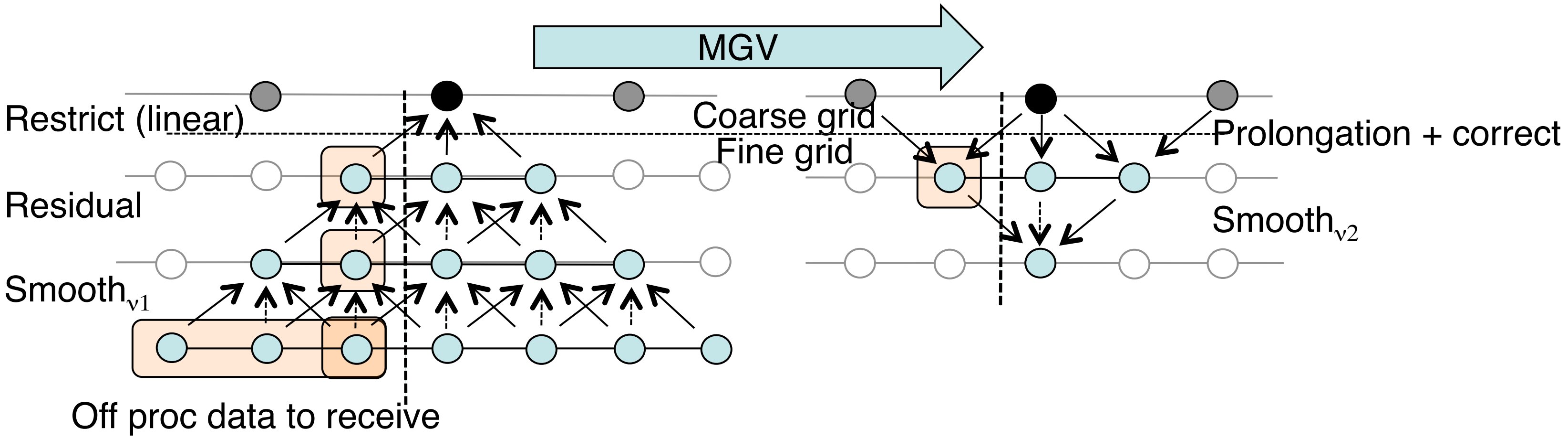
Cores	27	216	1,728	13,824
N (x10^6)	2.2	17.5	140	1,120
Solve Time	4.1	4.9	5.6	7.0
Setup (1)	5.2	6.1	13	28
S (2) partit.	9.2	11	21	155
Iterations	11	12	12	14
Mflops/s/ core	334	314	276	257

Outline

- Establish a lower bound on solver complexity
  - Apply ideas to Magnetohydrodynamics (MHD)
- Distributed memory & communication avoiding MG
  - Asynchronous unstructured Gauss-Seidel
- New algebraic multigrid (AMG) in PETSc
  - Application to 3D elasticity and 2D Poisson solves
- Data centric MG: cache aware & communication avoiding
  - Application to 2D 5-point stencil V(1,1) cycle

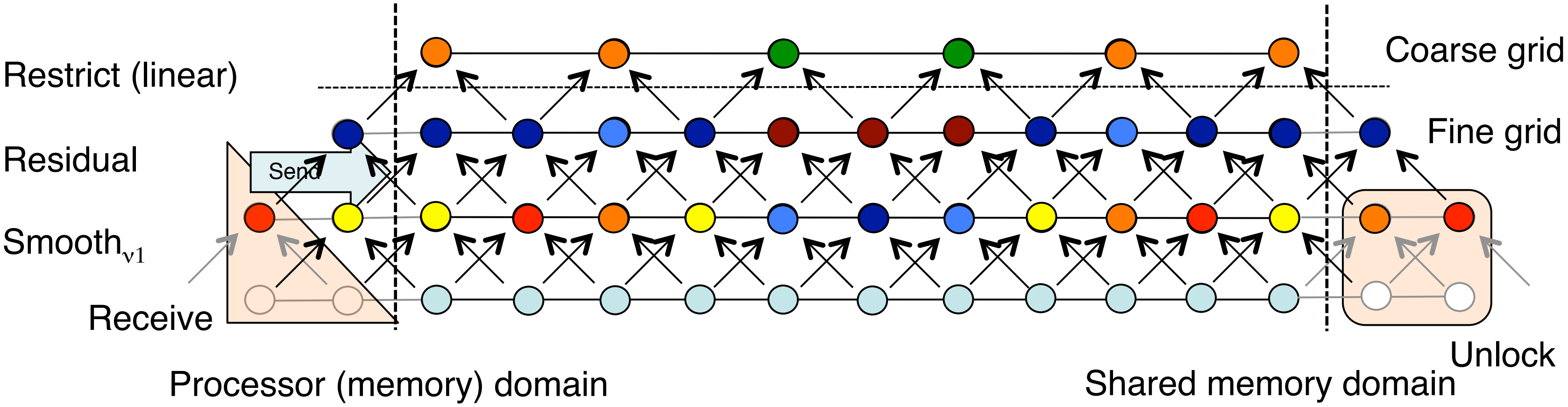
Data Centric Multigrid - V(1,1)

- MG algorithm: Sequential with parallel primitives
  - Common way to think and code.
- Problem: poor data reuse, low comp. intensity, much data movement
- A Solution: loop fusion (eg, C. Douglas et. al.)
  - “Vertical” partitioning of processing instead of (pure) “horizontal”
    - Vertex based method with linear restriction & prolongation
    - Fuse: one loop; course grid correction; 2nd loop
    - Data dependencies of two level MG,1D, 3-point stencil:



Hierarchical memory (cache & network) optimization - fusion

- Approach to fusing 1st leg of V-cycle, 1D, 3-point stencil
  - One smoothing step with simple preconditioner (ie, no new data dependencies)
  - Residual
  - Restriction
- Overlap communication and computation & aggregate messages *w/ multiple states*
  - Communication avoiding
- Multiple vectors (lhs, rhs, res, work) and vector ops (AXPY,etc.) not shown
- Arrows show data dependencies (vertical, self, arrows omitted)
- Processor domain boundary (left) w/ explicit message passing
- Shared memory domain (right) "unlocks" memory when available
- Boundary processing could be asynchronous
- Multiple copies of some data required (not shown) at boundaries and ghost regions

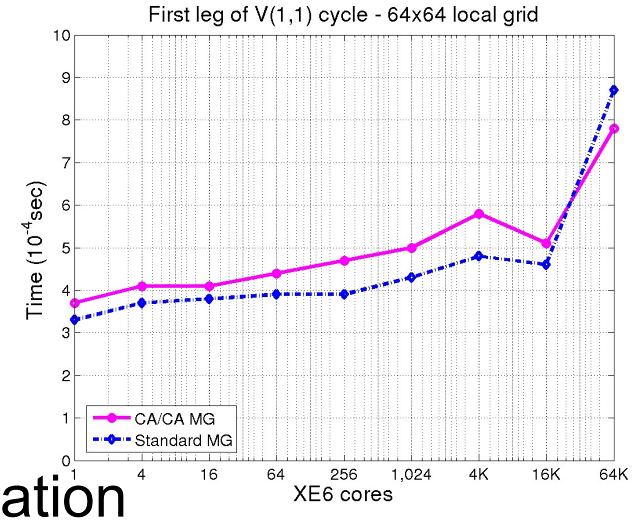


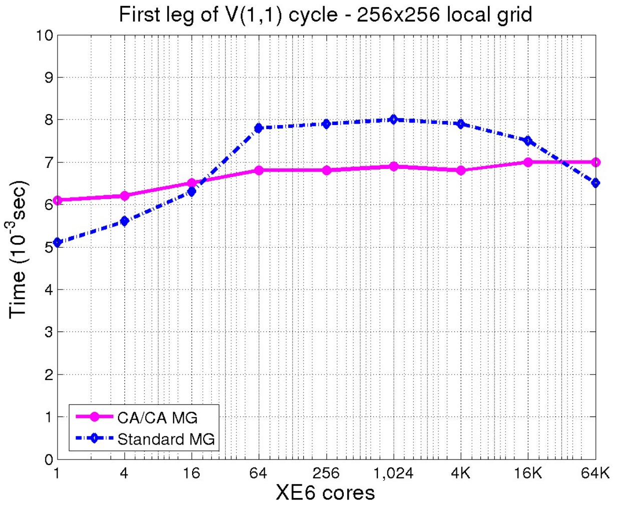
Multigrid V(ν_1,ν_2) with fusion

```
function u = MGV(A,f)
    If A coarsest grid
        $u \leftarrow A^{-1}f$
    else
        $u \leftarrow S^{\nu 1}(f, u)$                 -- Smoother (pre)
        $r \leftarrow f - Au$                          C. Douglas et.al.
        $r_H \leftarrow Rr$                            Chombo
        $e_H \leftarrow MGV( RAP, r_H )$               -- recursion (Galerkin)
        $u \leftarrow u + Pe_H$
        $u \leftarrow S^{\nu 2}(f, u)$                 -- Smoother (post)
```

Numerical tests

- Reference Implementation of first leg of V(1,1) cycle
  - 2D 5-point FV stencil
  - Linear interp./prol.
  - ~800 lines of FORTRAN
  - Horrible to code!
- Compare with standard implementation
  - Non-blocking send/recv
  - Overlap comm. & comp.
  - ~400 lines of FORTRAN
- Cray XE-6 at NERSC
  - Four levels of MG
  - 256 x 256 and 64 x 64 fine grid
- I am not a good compiler!



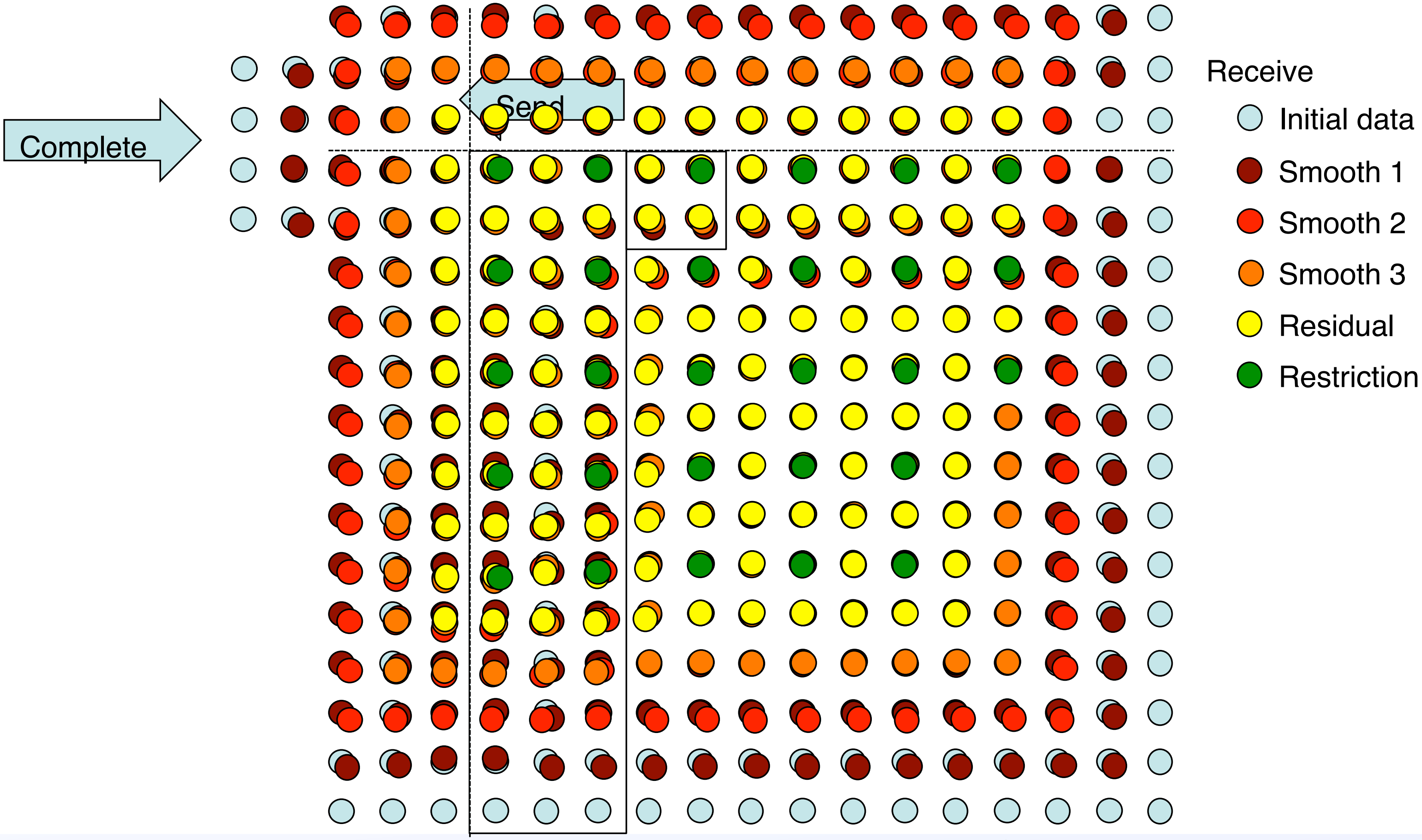


Conclusion

- Equations solvers are too big to fail
- Multigrid is a shovel ready algorithm
- Good distributed memory implementations are hard and getting harder with deep memory architectures
- Many-core node, data centric algorithms (loop fusion, GPUs,…) are not well suited to FORTRAN/C
- Need compiler/tools/language support
  - of some sort …

Thank you

2D, 9-point stencil,1st leg of V(3,3) w/ bilinear restriction



Complete

Send

Receive

○ Initial data

● Smooth 1

● Smooth 2

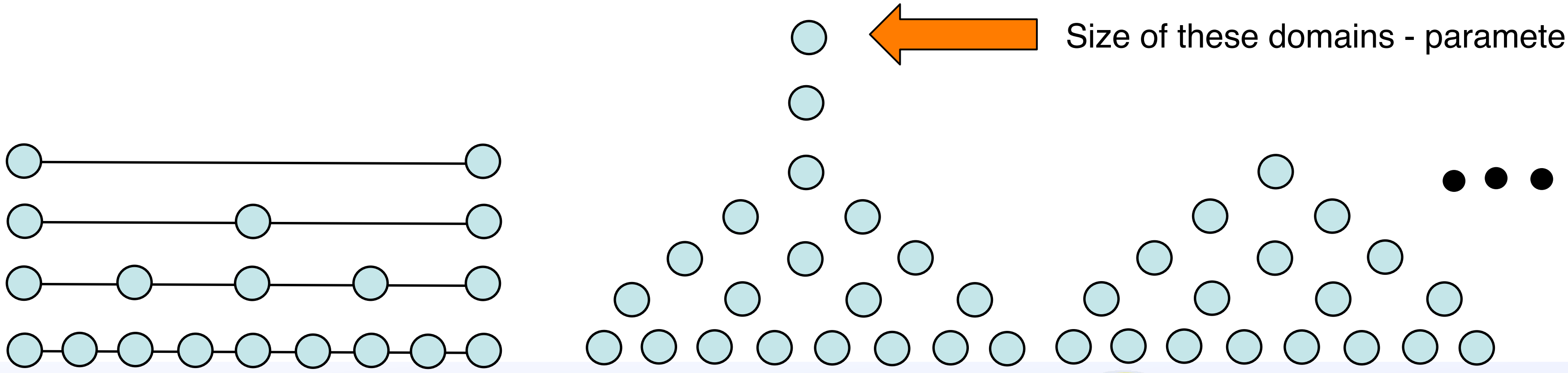
● Smooth 3

● Residual

● Restriction

A word about parallel complexity

- Solver work complexity:
  - M iterations * flops/iteration
  - All components of MG can have O(N) work complexity
    - Optimal – its takes O(N) work to print the solution
  - 1D C-cycle work complexity: C*N*(1+1/2+1/4+1/8…) < 2*C*N = O(N)
- Parallel complexity – work depth
  - V-cycle has O(log(N)) work depth
    - Optimal – Laplacian is fully coupled
      - ie, Green’s function has global support
    - Same as a dot product
  - F-cycles: O($\log^2$ (N))



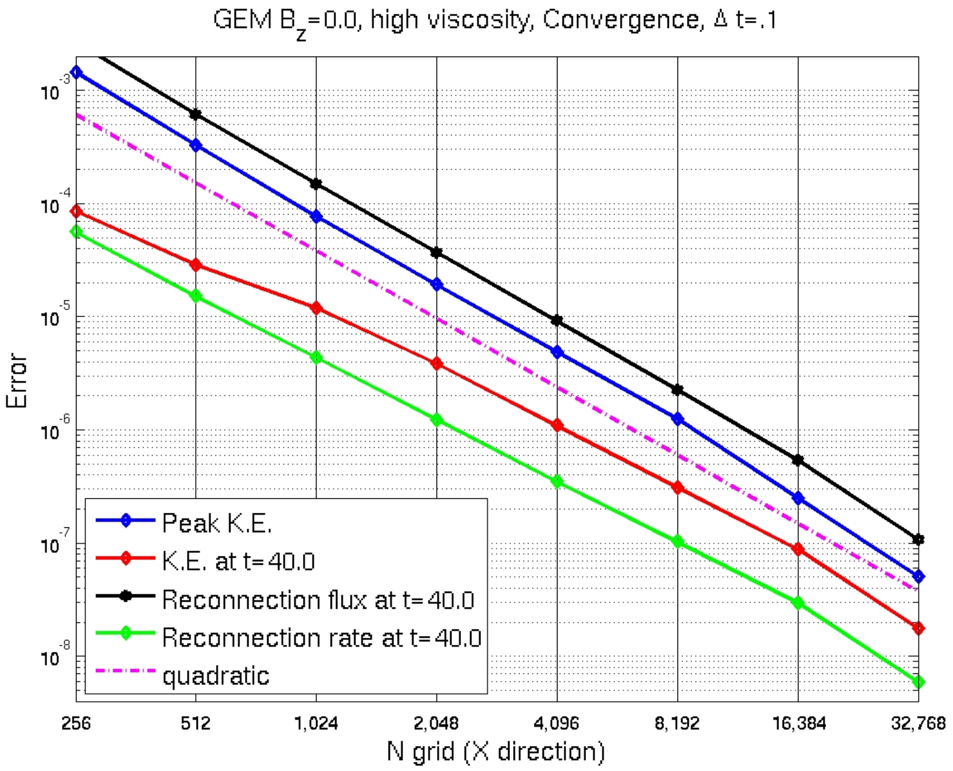
Size of these domains - parameter

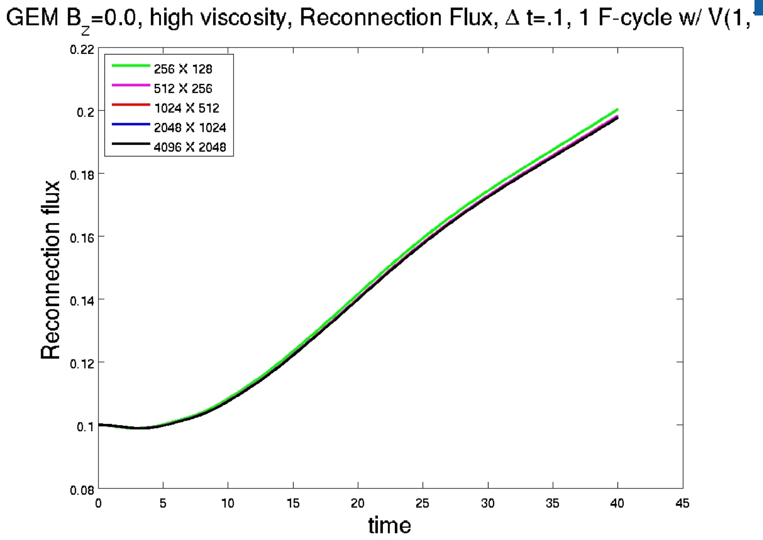
Solver Algorithm issues past and future

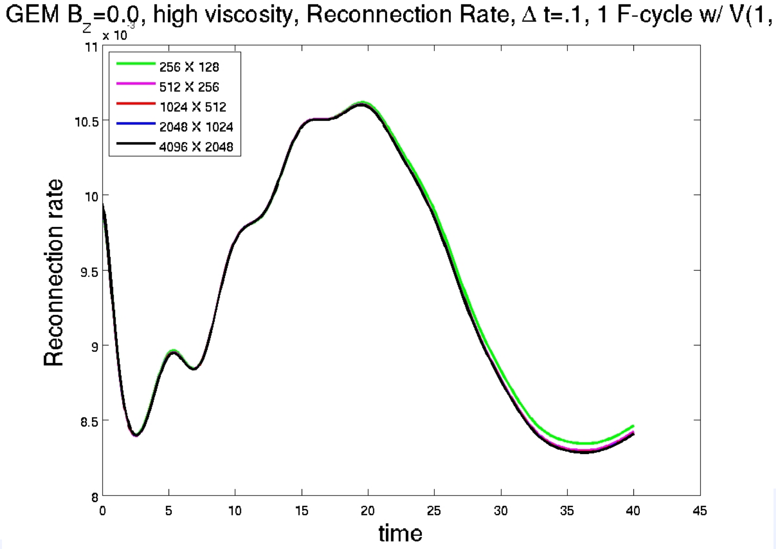
- Present and future: memory movement limited
- 70’s had similar problems as today, and what we see as the future
  - Then: couldn’t afford memory – matrix free
  - Now: can’t afford to architect it and use it
- 80’s were pernicious:
  - Ubiquitous uniform access memory and big hair …
  - Big memory did allow AMG and direct solvers to flourish
- Solutions that work on exa-scale machines … look to the 70’s
  - Low memory, matrix free, algorithms
  - Perhaps more regular grids as well
- Multigrid can solve with spatial/incremental truncation error accuracy
  - With work complexity of as low as ~6 residual calculations (work units)
  - On the model problem: low order discretization of Laplacian
    - Proven 30 years ago
  - “Textbook” multigrid efficiency
- No need to compute a residual (*no synchronous* norm computations)
- *No* need for CG’s *synchronous* dot products
- MG is weakly synchronized but this comes from the elliptic operator complexity
  - no way around it
- MG has O(N) work complexity in serial, O(log(N)) work depth in parallel
  - F-cycles, required for truncation accurate solutions, is O($\log^2$(N))
- Work complexity looks less relevant now – “memory movement” complexity?

Verify 2nd order convergence

- 2nd order spatial accuracy
  - Achieved with F-cycle MG solve
- $B_z = 0$, high viscosity
- Up to 1B cells (8B equations)


GEM B_z=0.0, high viscosity, Convergence, Δ t=.1


GEM B_z=0.0, high viscosity, Reconnection Flux, Δ t=.1, 1 F-cycle w/ V(1,


GEM B_z=0.0, high viscosity, Reconnection Rate, Δ t=.1, 1 F-cycle w/ V(1,

Multigrid performance - smoothers

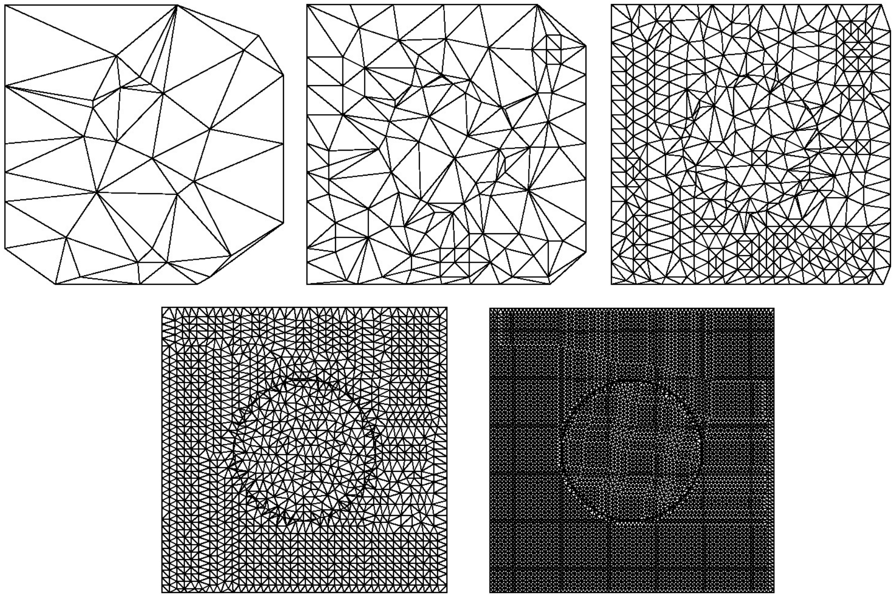
- Multigrid splits the problem into two parts
  - Coarse grid functions (MG method proper) - takes care of scaling
  - Smoother (+ exact coarse grid solver) - takes care of physics
- Smoothers, where most of flops are – important for performance opt.
- Additive MG smoother requires damping
  - $Be = (I – (B_1 + B_2 + \ldots B_m)A)e$
  - Good damping parameter not always available
    - eg, non-symmetric problems
  - Krylov methods automatically damp
    - But not stationary & have hard synchronization points
- Multiplicative smoothers (eg, Gauss-Seidel)
  - $Be = (I – B_1A) (I – B_2A) \ldots (I – B_mA)e$
  - Excellent MG smoother in theory
  - Distributed memory algorithm is a hard problem
    - Exploit nature of FE/FD/FV graphs …

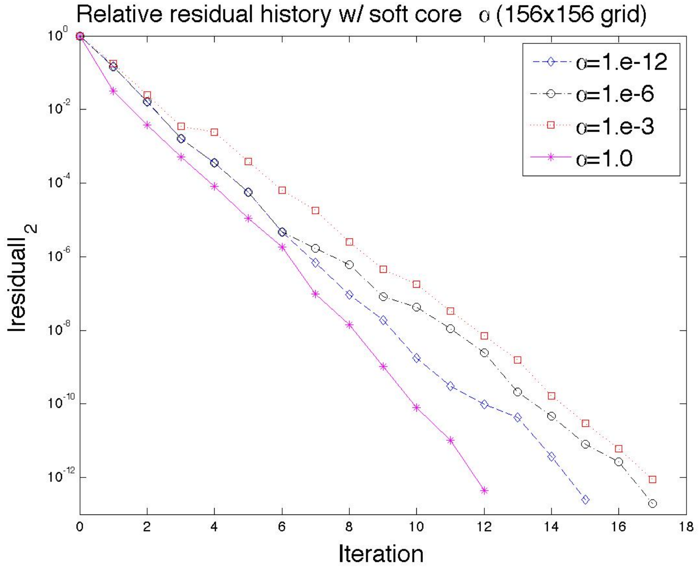
Common parallel primitives for AMG

- Matrix matrix products:
  - $A_{i+1} = P^T A_i P$
  - $P = (I - \omega D^{-1} A) P_0$
- Computing (re)partitioning (ParMetis)
- Moving matrices (repartitioning)
- Maximal Independent Sets of A^k - MIS(k)
  - Useful mechanism for aggregation
  - Want coarsening factor of about 3
    - This is perfect on regular hexahedra mesh

Unstructured geometric multigrid

- Select coarse points
  - MIS(1)
- Remesh (TRIANGLE)
- Use finite element shape functions for restriction/ prolongation
- Example: 2D square scalar Laplacian with “soft” circle





Coarse grid complexity at extreme scales

- Multigrid has theoretically optimal parallel complexity
  - “Data movement” complexity?
- Log(N) computational depth - not enough parallelism available on coarse grids
- Coarse grid complexity is main source of inefficiency at extreme scales
- AMG issues: Support of coarse grid functions tend to grows
  - Independent sets are useful in coarsening
    - Independent set: set of vertices w/o edges between each other
    - Maximal: can not add a vertex and still be independent
  - The *maximum* independent set give 3^3 (27) aggs, every 3rd point on 3D cart. grid
    - This is perfect for SA - no support growth on coarse grids & recovers geo. MG
  - But support grows on unstructured problems, for example consider
    - stencil grows from 27 to 125 points (extra layer)
    - One vertex/proc – communicate with ~124 procs
    - 3^3 vertex/proc – communicate with ~26 procs
- Thus, coarse grid memory complexity increases communication
- Amelioration strategy: use same basic idea as in parallel G-S:
  - Keep processor sub-domains from getting tiny (at least a few “stencils”)
  - Reduce active processors (eg, keep ~500 equations per processor)
    - This leads to need to repartition if original data was not recursively partitioned
    - No data locality with randomly aggregating sub-domains