4-5-2021

Lecture 06: The Impact of Computer Architectures on the Design of Algebraic Multigrid Methods

Ulrike Yang
Lawrence Livermore National Laboratory

The Impact of Computer Architectures on the Design of Algebraic Multigrid Methods

Ulrike Meier Yang

April 5, 2021

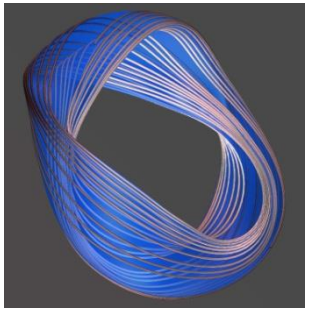
Spring Lecture Series, Univ of Arkansas

LLNL-PRES-821025
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

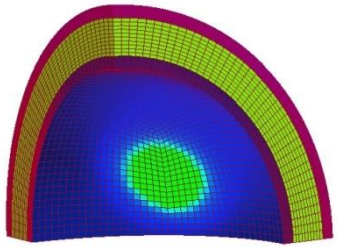
Lawrence Livermore National Laboratory

Background and Motivation

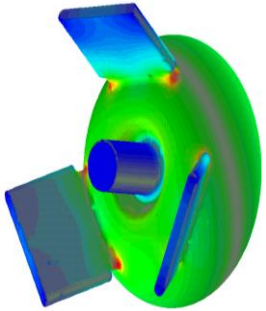
- The solution of linear systems is at the core of many scientific simulation codes



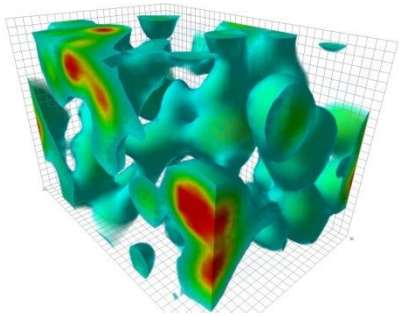
Magnetohydrodynamics



Elasticity / Plasticity



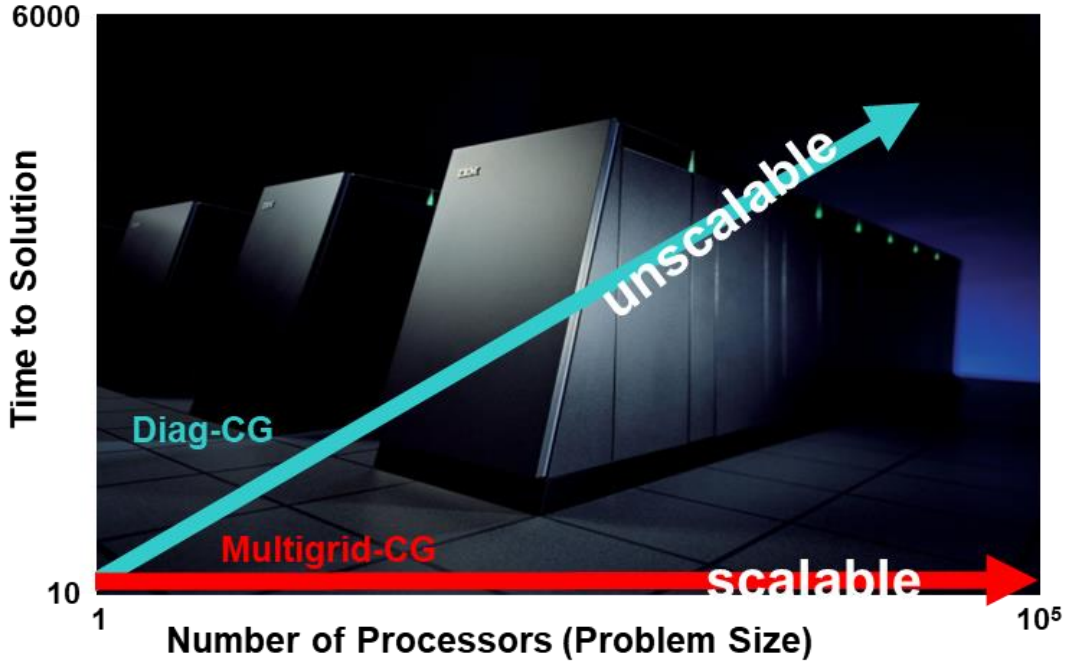
Electromagnetics



Quantum Chromodynamics

- High fidelity requires huge linear systems and large-scale computing
- Development of parallel linear solvers and software (*hypre*), driven by applications, with focus on multigrid methods

Multigrid is playing an important role for high performance computing



- Multilevel methods are optimal, i.e., O(N) operations
- Well designed multilevel methods have excellent scalability
- Scalable → faster simulations
  → better science!

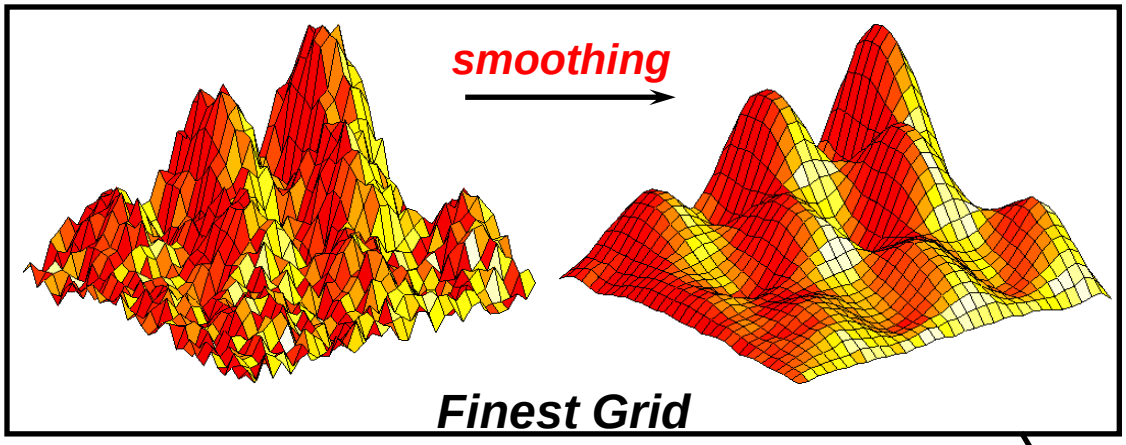
Algebraic multigrid has been impacted by rapidly changing computer architectures over the years!

- New architectures impact algorithm design, requiring consideration of:
  - **Parallelism:**
    - Vector computers (e.g., Cray X-MP)
    - Distributed computer architectures
  - **+ avoiding/reducing communication and memory movement:**
    - Computers with up to millions of cores (e.g., IBM Blue Gene series)
    - Hybrid computer architectures with very fast cores, complicated memory hierarchies
  - **+ extreme-scale parallelism:**
    - Heterogeneous computer architectures with accelerators/GPUs (e.g., Roadrunner, LANL, Summit, ORNL)

High Performance Computing – the race is on!

- Greater than 500,000x increase in supercomputer performance, with no end currently in sight!
- We are now pursuing exascale computing!
  - Exaflop = 10^{18} calculations per second
  - Fugaku first computer to achieve this using ARM chip technology
  - Other exascale systems on the horizon using GPU technology:
    - Frontier (Cray/AMD, ORNL, 2021/2022)
    - Aurora (Cray/Intel, ANL, 2021/2022)
    - El Capitan (Cray/AMD, LLNL, 2023)

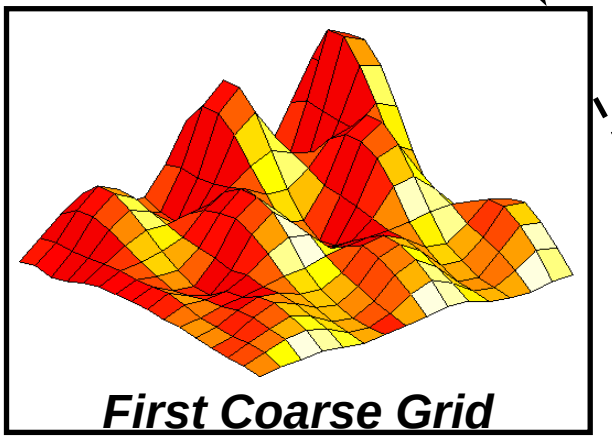
Multigrid uses coarse grids to efficiently damp out smooth error components



The Multigrid V-cycle

restriction

prolongation (interpolation)



Note: smaller grid

Background of Algebraic Multigrid

- Algebraic multigrid (AMG) formally based on geometric multigrid method, looks similar
- Originally developed by Brandt, McCormick, Ruge (1982)
- AMG developed to deal with more complex problems, unstructured problems
- Deal with variables instead of grids
- Efficient interplay between smoothing and coarse-grid correction

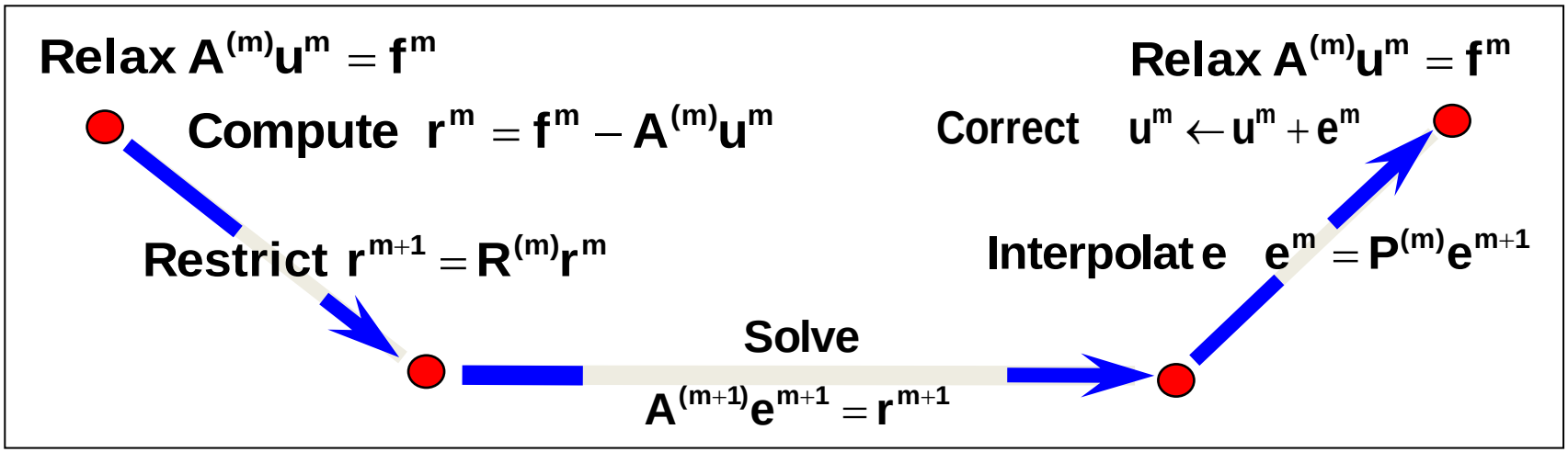
Algebraic Multigrid

Setup Phase:

- Select coarse "grids,"
- Define interpolation, $P^{(m)}, \quad m = 1,2,\ldots$
- Define restriction and coarse-grid operators

$$R^{(m)} (= P^{(m)T}) \qquad A^{(m+1)} = R^{(m)} A^{(m)} P^{(m)}$$

Solve Phase:



Parallelizing AMG

- Multigrid linear solvers are optimal ($O(N)$ operations), and hence have good scaling potential
- Many of AMG's components are parallel, e.g., Galerkin product RAP, interpolation operator generation, matrix-vector operations
- Some critical components are highly sequential, e.g., smoothers (lex. Gauss-Seidel), coarsening algorithm
- We consider here a distributed memory model, i.e., message passing (MPI) programming model

Original Smoother: Gauss-Seidel – sequential!

Problem:

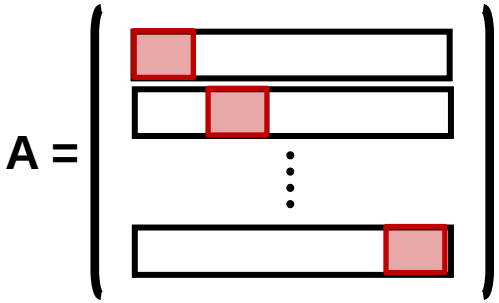
- solving: **Ax=b, A** symmetric positive definite
- smoothing: $x_{n+1} = x_n + M^{-1}(b - A x_n)$

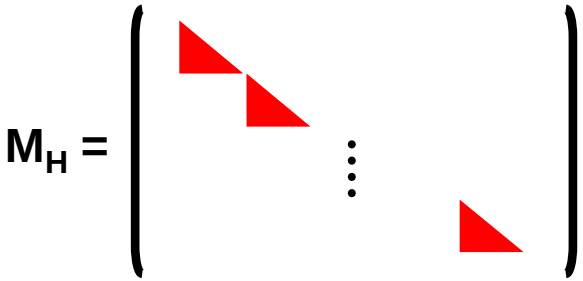
Weighted Jacobi smoother: $M = \omega D$

Hybrid Gauß-Seidel smoothers: $M = M_H$

- Depends on number of processes
- Might include a weight for better convergence
  - $\omega = \frac{1}{\rho(M_H^{-1}A)}$

  But requires eigenvalue estimate





UMY, NLAA, 11 (2004), pp. 155-172.

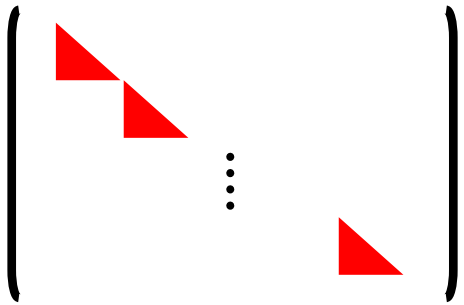
We pursue two strategies:
1. investigate alternate smoothers , 2. 'fix' hybrid smoothers

Alternate smoothers: **polynomial smoother** $I - M^{-1}A = p(A), p(0)=1$

- independent of parallelism
- MatVec kernel has been tuned
- But need extreme eigenvalue estimates

'Fix' hybrid smoothers: ℓ_1 **– smoother**

- Add suitable diagonal matrix to hybrid smoother M_H
  $M_{\ell_1} = M_H + D_{\ell_1}$ with $D_{\ell_1} = \sum_{j \neq i} |a_{ij}^o|$
- Always convergent, when GS converges
- Requires no eigenvalue estimates

$M_H =$ 

Baker, Falgout, Kolev, UMY, SISC, 33 (2011), pp. 2864-2887 .

Choosing the coarse grid

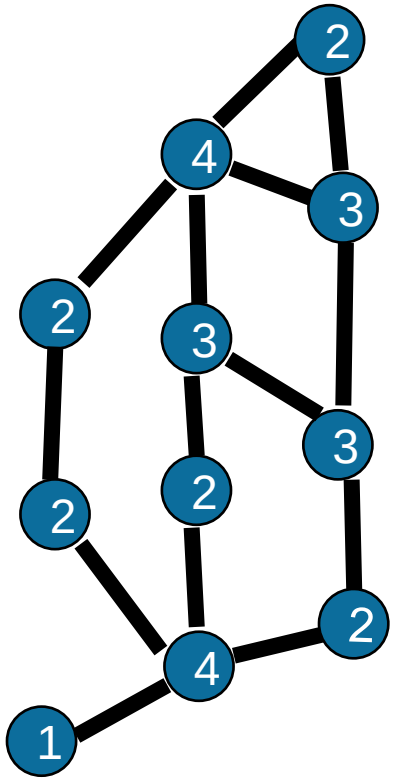
- Focus here on
  Classical AMG (C-AMG) – coarse grid is a subset of the fine grid

- The basic coarsening procedure is as follows:
  - Define a strength matrix A_s by deleting weak connections in A
  - First pass: Choose an independent set of fine-grid points based on the graph of A_s
  - Second pass: Choose additional points if needed to satisfy interpolation requirements

- Coarsening partitions the grid into *C*- and *F*-points

Ruge, Stüben, Algebraic multigrid (AMG), in : S. F. McCormick, ed., Multigrid Methods, vol. 3 of Frontiers in Applied Mathematics (SIAM, Philadelphia, 1987) 73–130.

Original AMG coarsening highly sequential!

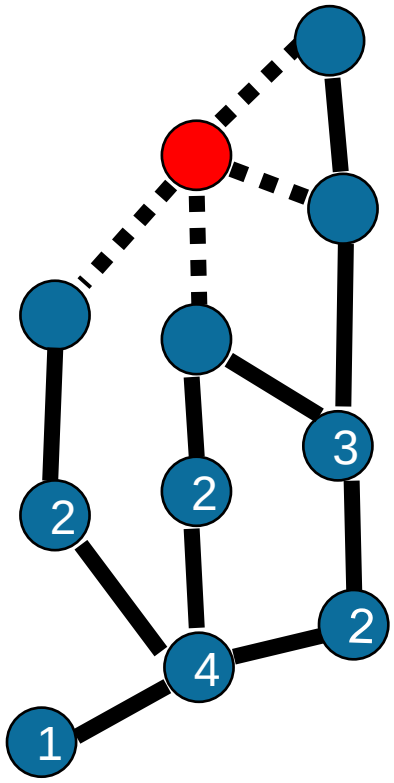


- (C1) Maximal Independent Set:

Independent: no two C-points are connected

Maximal: if one more C-point is added, the independence is lost

Original AMG coarsening highly sequential!

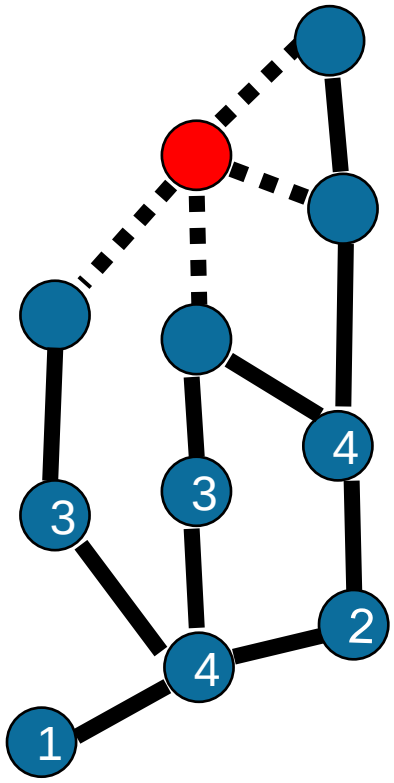


- (C1) Maximal Independent Set:

Independent: no two C-points are connected

Maximal: if one more C-point is added, the independence is lost

Original AMG coarsening highly sequential!

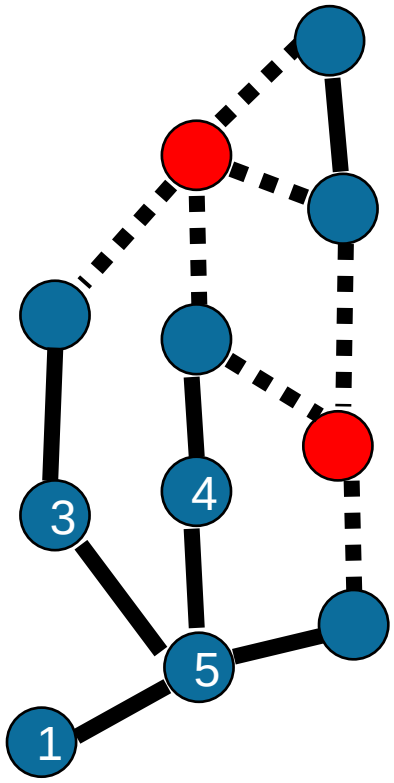


- (C1) Maximal Independent Set:

Independent: no two C-points are connected

Maximal: if one more C-point is added, the independence is lost

Original AMG coarsening highly sequential!

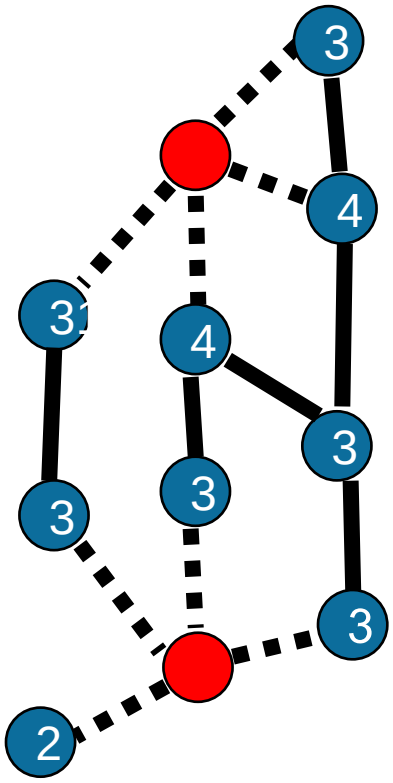


- (C1) Maximal Independent Set:

Independent: no two C-points are connected

Maximal: if one more C-point is added, the independence is lost

Original AMG coarsening highly sequential!

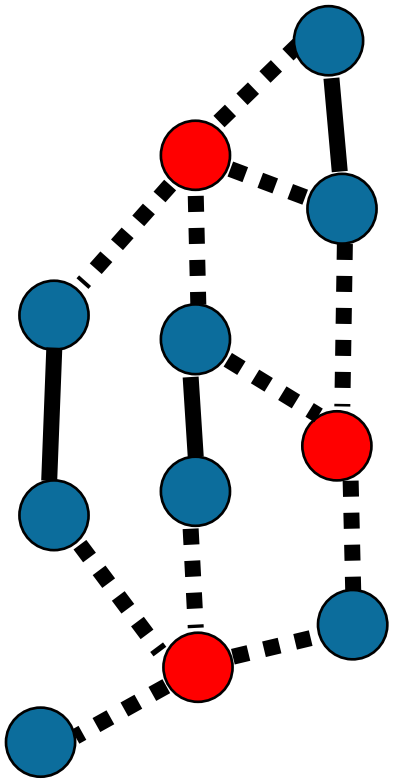


- (C1) Maximal Independent Set:

Independent: no two C-points are connected

Maximal: if one more C-point is added, the independence is lost

Original AMG coarsening highly sequential!

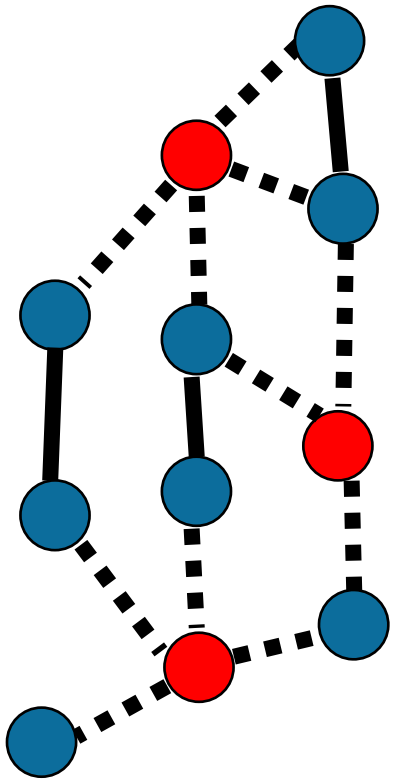


- (C1) Maximal Independent Set:

Independent: no two C-points are connected

Maximal: if one more C-point is added, the independence is lost

Original AMG coarsening highly sequential!



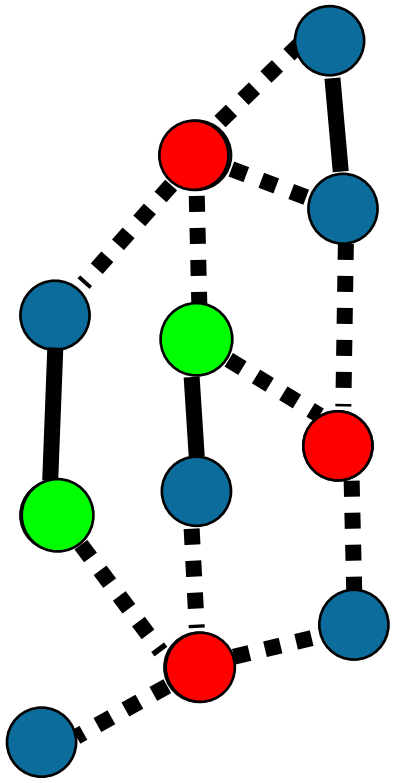
- (C1) Maximal Independent Set:

Independent: no two C-points are connected

Maximal: if one more C-point is added, the independence is lost

- (C2) All F-F connections require connections to a common C-point (for good interpolation)

Original AMG coarsening highly sequential!



- (C1) Maximal Independent Set:

Independent: no two C-points are connected

Maximal: if one more C-point is added, the independence is lost

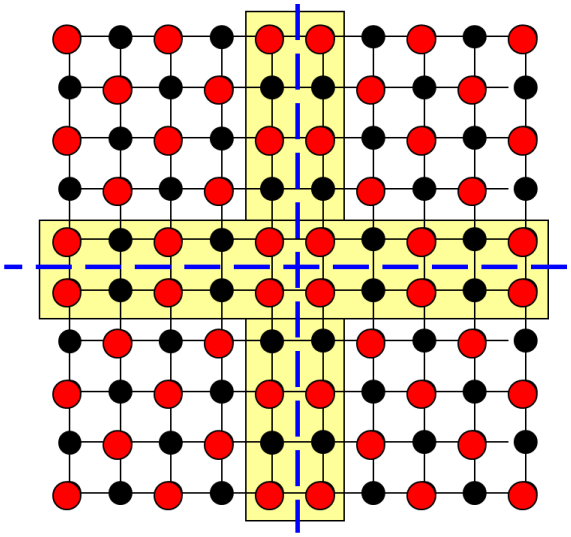
- (C2) All F-F connections require connections to a common C-point (for good interpolation)

- *F*-points have to be changed into *C*-points, to ensure (C2); (C1) is violated

more C-points, higher complexity

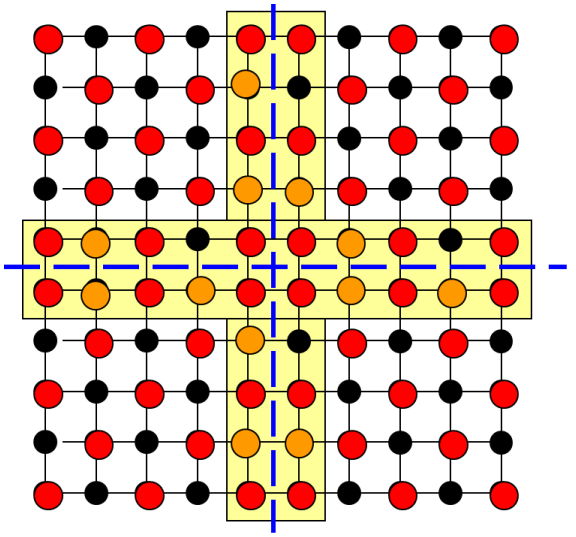
Introduce parallel coarsening algorithms

- Use sequential algorithm within each process
- Find a way on how to deal with the process boundaries:
  - Do nothing and hope it works
    - Con: risky, might miss important connections



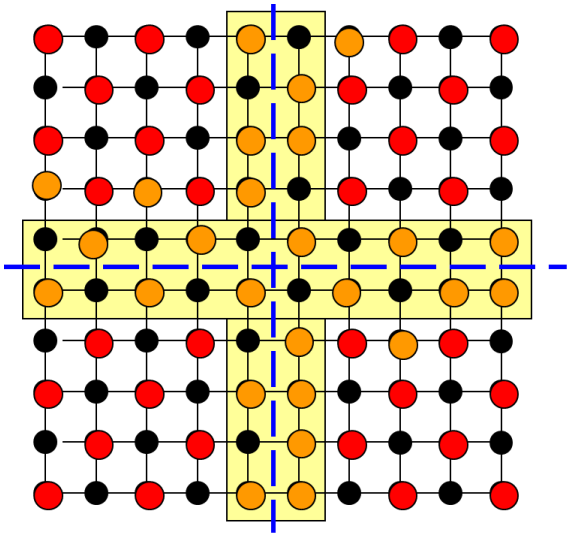
Introduce parallel coarsening algorithms

- Use sequential algorithm within each process
- Find a way on how to deal with the process boundaries:
  - Do nothing and hope it works
    - Con: risky, might miss important connections
  - Perform a third path on the boundary points
    - Con: can lead to too many C-points



Introduce parallel coarsening algorithms

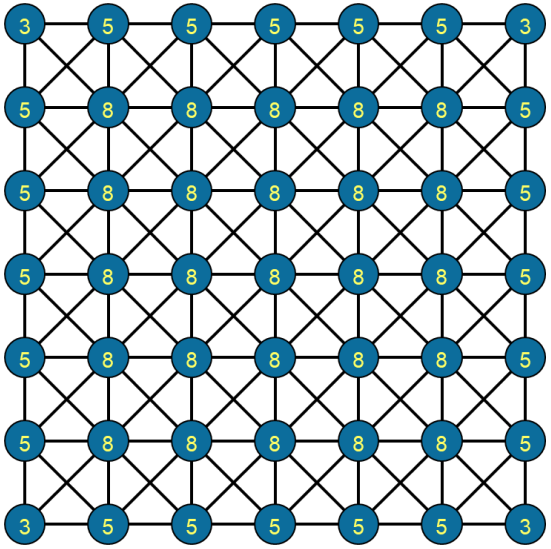
- Use sequential algorithm within each process
- Find a way on how to deal with the process boundaries:
  - Do nothing and hope it works
    - Con: risky, might miss important connections
  - Perform a third path on the boundary points
    - Con: can lead to too many C-points
  - Apply the parallel CLJP algorithm (introduced on the next slide) to the process boundary points only: Falgout coarsening
    - Best solution for this scenario



Henson, UMY, Appl. Num. Math, 41 (2002), pp. 155-177.

CLJP Coarsening Algorithm

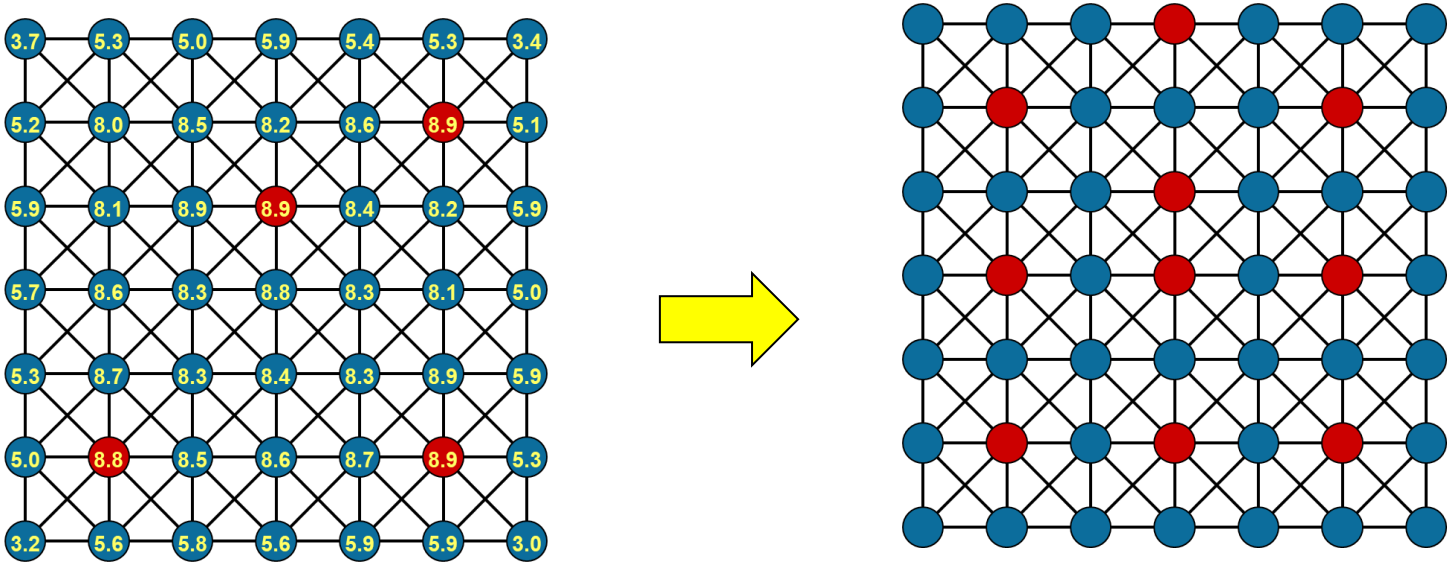
- CLJP (Cleary-Luby-Jones-Plassmann) coarsening
- Based on an idea by Cleary[3] and algorithms by Luby[1], Jones and Plassmann[2]
- Uses one-pass approach with random numbers to get concurrency



[1] Luby, Journal on Computing 15 (1986) 1036–1053.
[2] Jones and P. E. Plassman, A parallel graph coloring heuristic, SISC, 14 (1993) 654–669.
[3] Cleary, Falgout, Henson, Jones, Proceedings (Springer-Verlag, New York, 1998).

CLJP Coarsening Algorithm

- CLJP (Cleary-Luby-Jones-Plassmann) coarsening
- Based on an idea by Cleary[3] and algorithms by Luby[1], Jones and Plassmann[2]
- Uses one-pass approach with random numbers to get concurrency



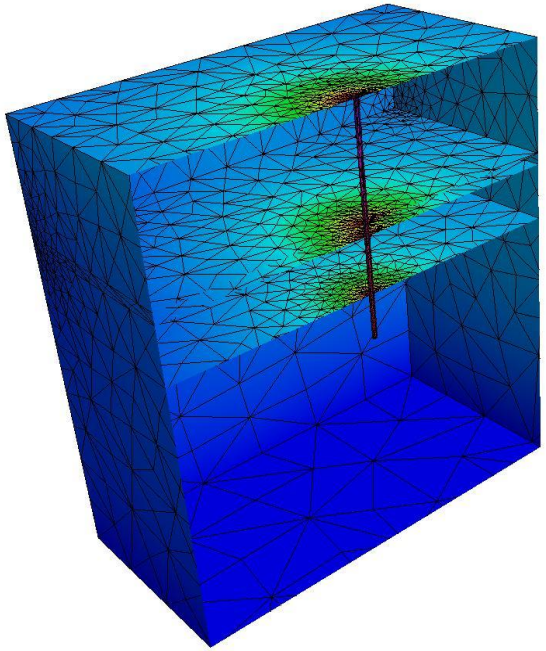
[1] Luby, Journal on Computing 15 (1986) 1036–1053.
[2] Jones and P. E. Plassman, A parallel graph coloring heuristic, SISC, 14 (1993) 654–669.
[3] Cleary, Falgout, Henson, Jones, Proceedings (Springer-Verlag, New York, 1998).

Unstructured Diffusion Problem with Jumps

- AMG complexity growth for large 3-dimensional problems

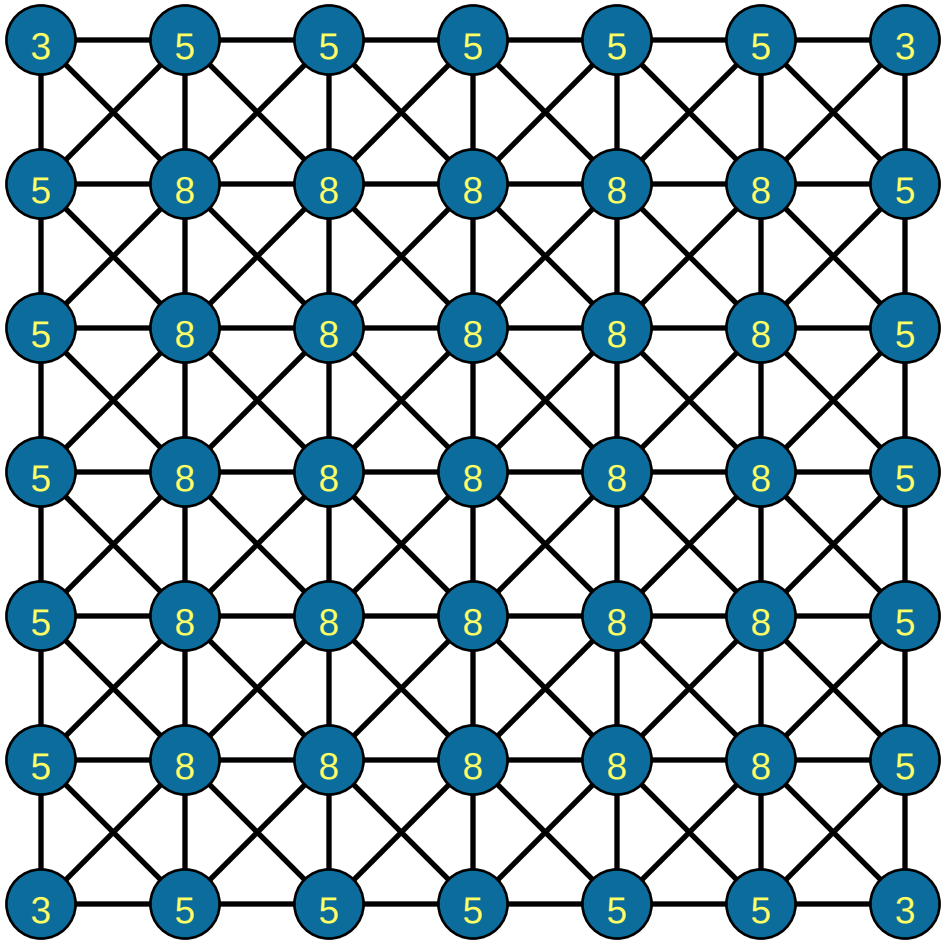
Operator complexity

$$\text{o.c.} = \frac{\sum_i nnz(A^{(i)})}{nnz(A)}$$



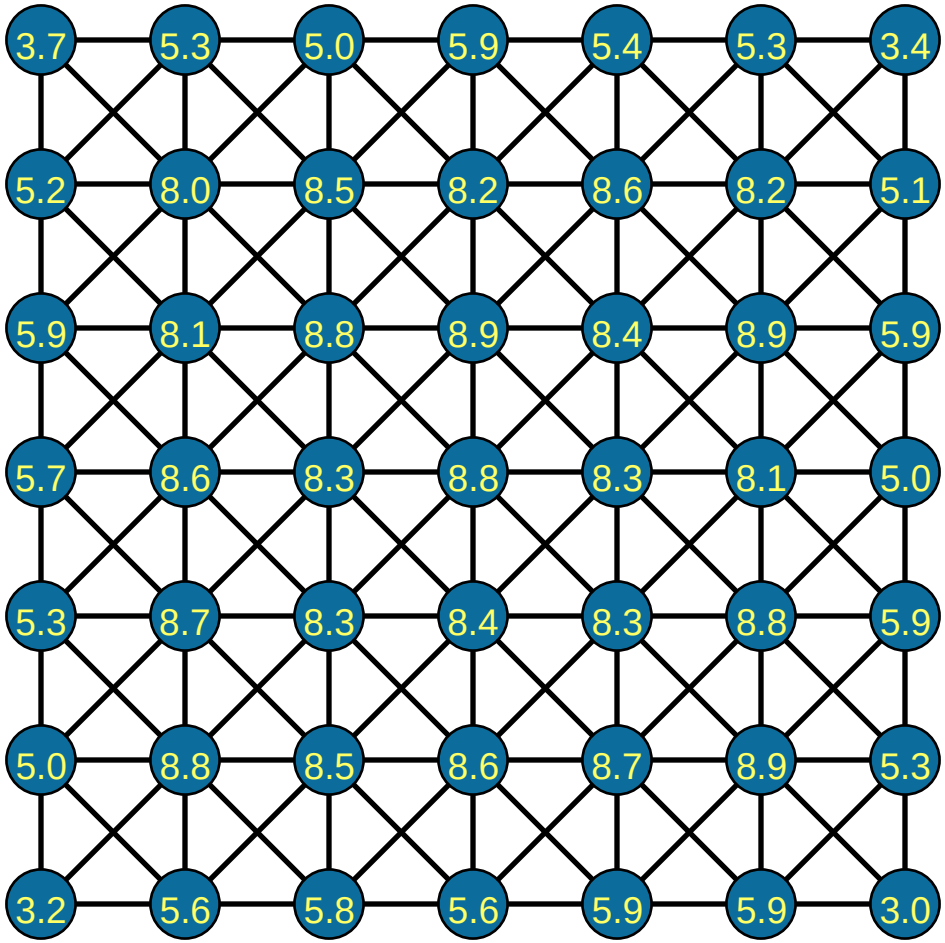
#procs	CLJP		
	#its	o.c	time
1	**9**	**5.6**	11
8	**9**	**6.5**	21
64	**11**	**6.7**	38
128	**10**	**7.9**	57
256	**11**	**7.8**	76
512	**11**	**7.2**	128
1024	**10**	**8.6**	210

PMIS: start



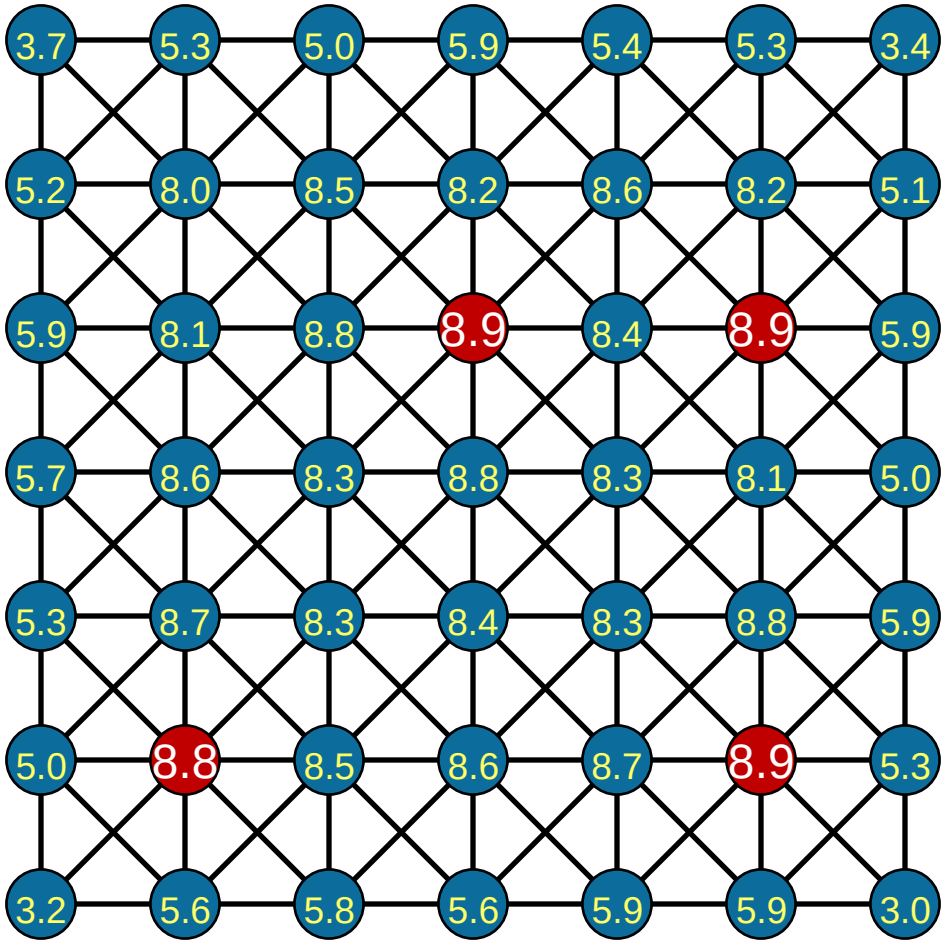
- select C-pt with maximal measure
- select neighbors as F-pts

PMIS: add random numbers



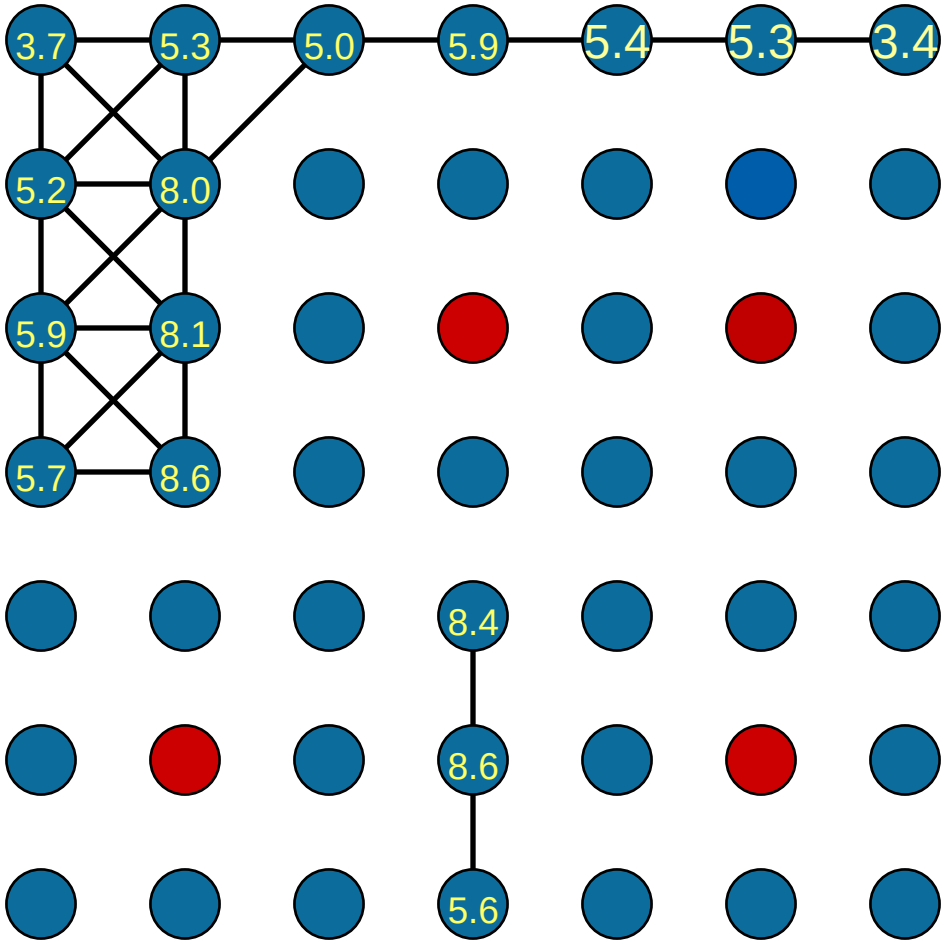
- select C-pt with maximal measure
- select neighbors as F-pts

PMIS: select



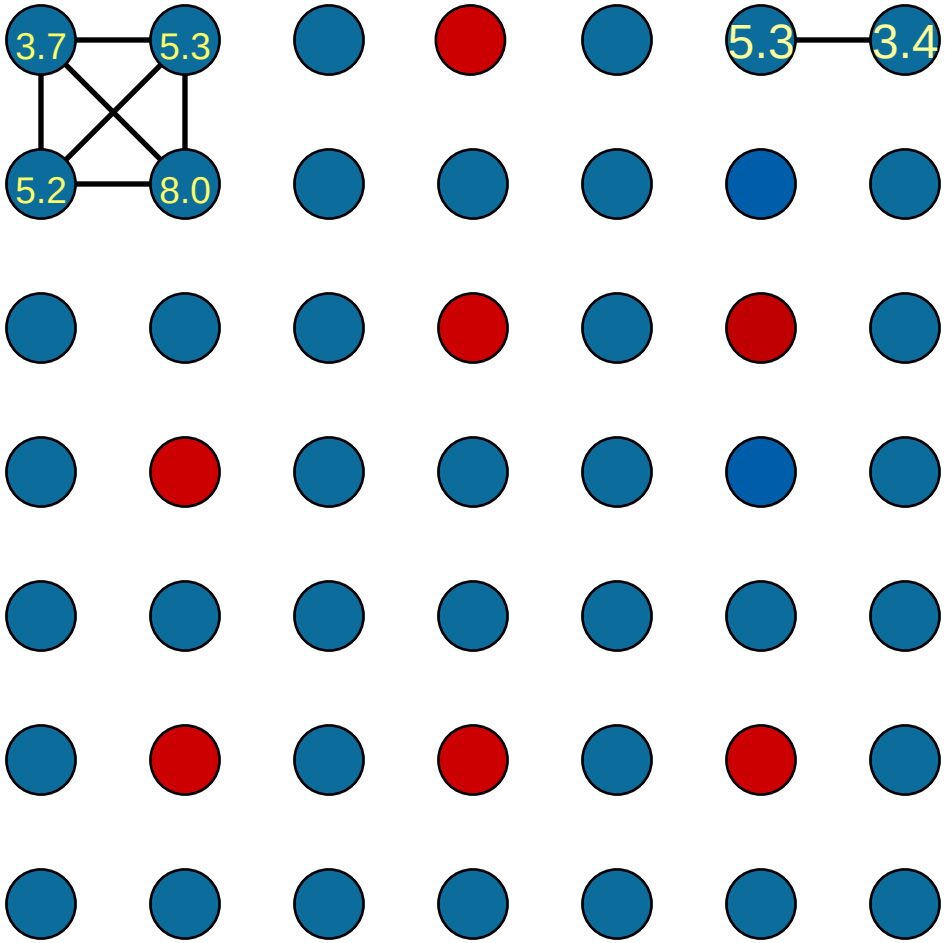
- select C-pts with maximal measure locally
- make neighbors F-pts

PMIS: update 1



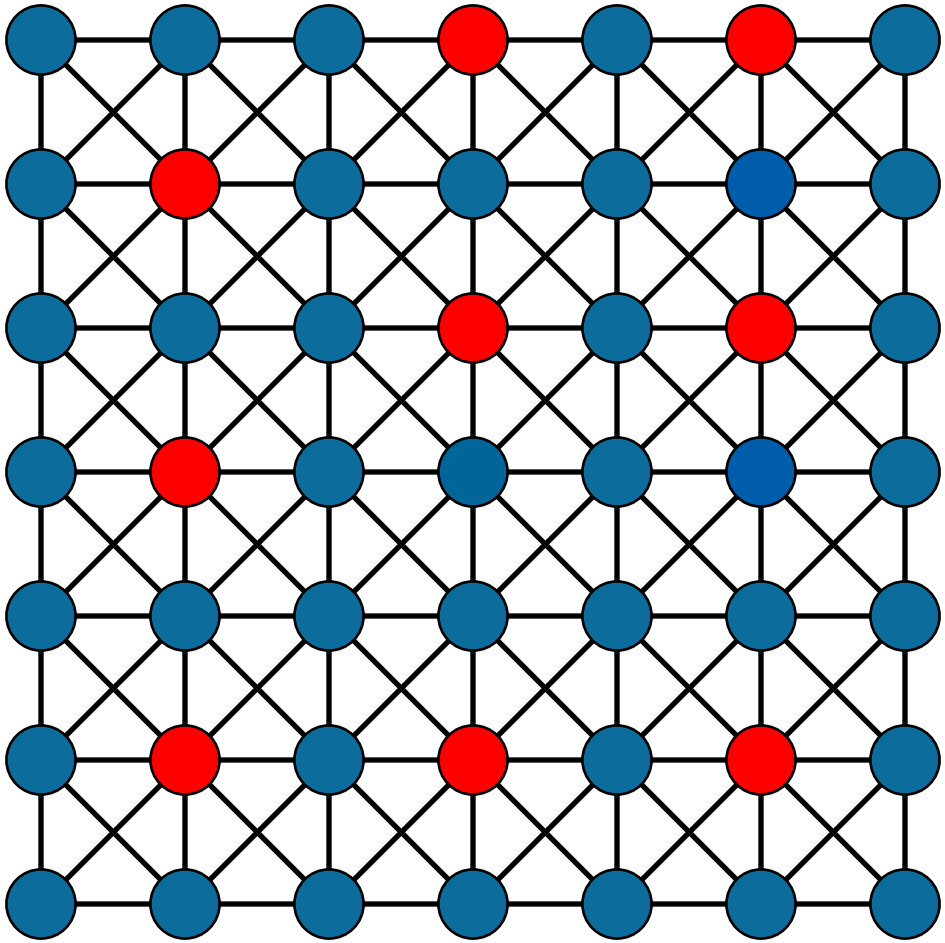
- select C-pts with maximal measure locally
- make neighbors F-pts (requires neighbor info)

PMIS: update 2



- select C-pts with maximal measure locally
- make neighbors F-pts

PMIS: final grid



- select C-pts with maximal measure locally
- make neighbor F-pts
- remove neighbor edges

Unstructured Diffusion Problem with Jumps

- Leads to significantly lower complexities and better times, but decreases numerical scalability

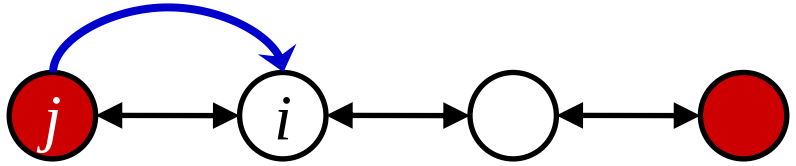
#procs	CLJP			PMIS		
	#its	o.c	time	#its	o.c.	time
1	9	5.6	11	**18**	**1.5**	**7**
8	9	6.5	21	**30**	**1.6**	**16**
64	11	6.7	38	**62**	**1.5**	**29**
128	10	7.9	57	**64**	**1.5**	**31**
256	11	7.8	76	**72**	**1.5**	**27**
512	11	7.2	128	**118**	**1.6**	**55**
1024	10	8.6	210	**162**	**1.5**	**77**

De Sterck, UMY, Heys, SIMAX, 27 (2006), pp. 1019-1039.

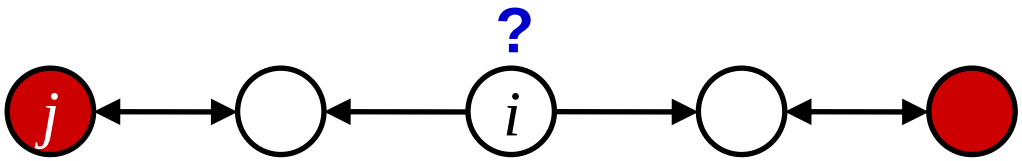
C-AMG interpolation is not suitable for more aggressive coarsening

- PMIS is parallel and eliminates the second pass, which can lead to the following scenarios:

One-sided interpolation



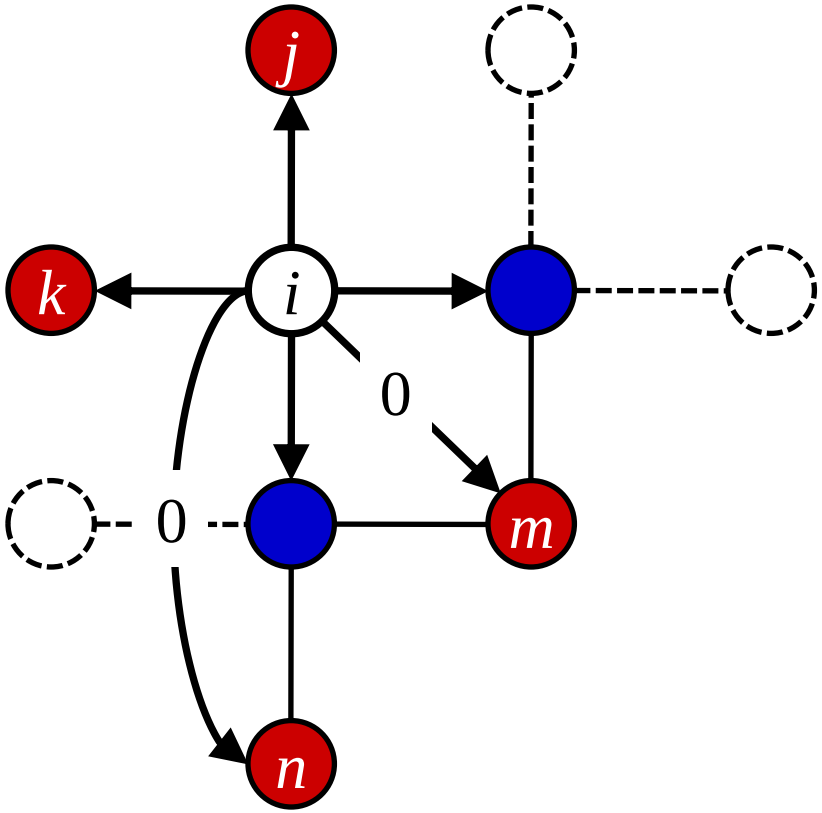
No interpolation



- Want above *i*-points to interpolate from both *C*-points
- Long-range (distance two) interpolation!

One possibility for long-range interpolation is extended interpolation

- C-AMG: $C_i = \{j,k\}$
- Long-range: $C_i = \{j,k,m,n\}$
- Extended interpolation – apply C-AMG interpolation to an extended stencil
- Extended+i interpolation is the same as extended, but also collapses to point i
- Improves overall quality
- Note that this requires a 2nd layer of communication!



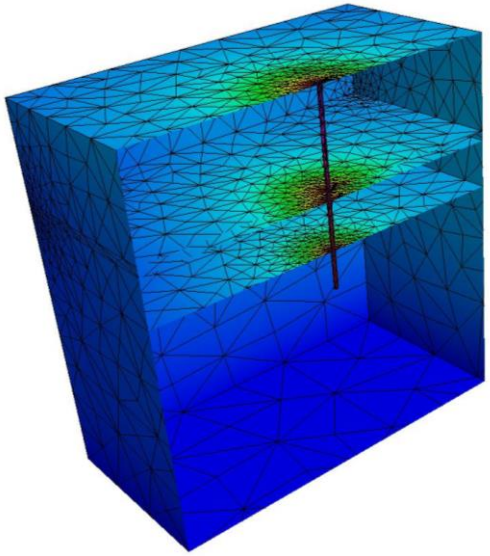
De Sterck, Falgout, Nolting, UMY, NLAA 15 (2008), pp. 115-139.

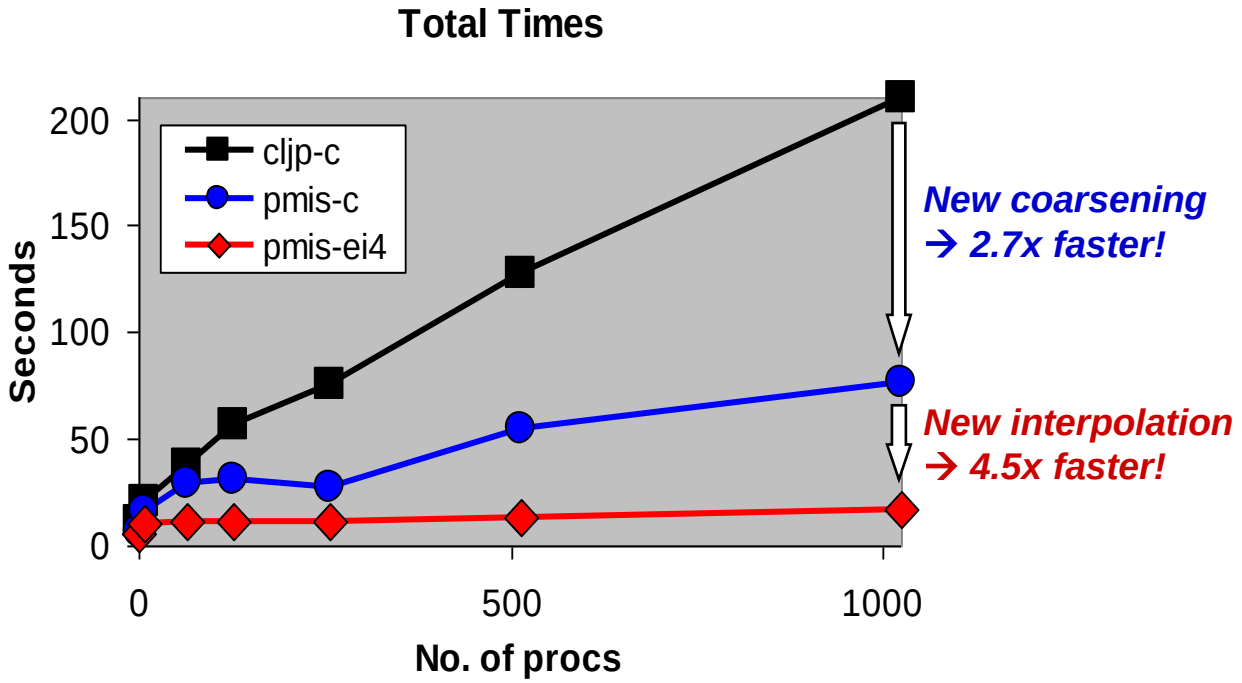
Unstructured Diffusion Problem with Jumps

#procs	CLJP		PMIS			
	class		class		e+i(4)	
	its	Op.c	its	Op.c	its	Op.c
1	9	5.6	18	1.5	**9**	**1.8**
8	9	6.5	30	1.6	**11**	**1.9**
64	11	6.7	62	1.5	**13**	**1.9**
128	10	7.9	64	1.5	**12**	**1.9**
256	11	7.8	72	1.5	**13**	**1.8**
512	11	7.2	118	1.6	**12**	**1.8**
1024	10	8.6	162	1.5	**14**	**2.0**

Note that full distance-two interpolation can still lead to large complexities and generally, is combined with interpolation truncation

Our new long-range interpolation approaches are improving scalability



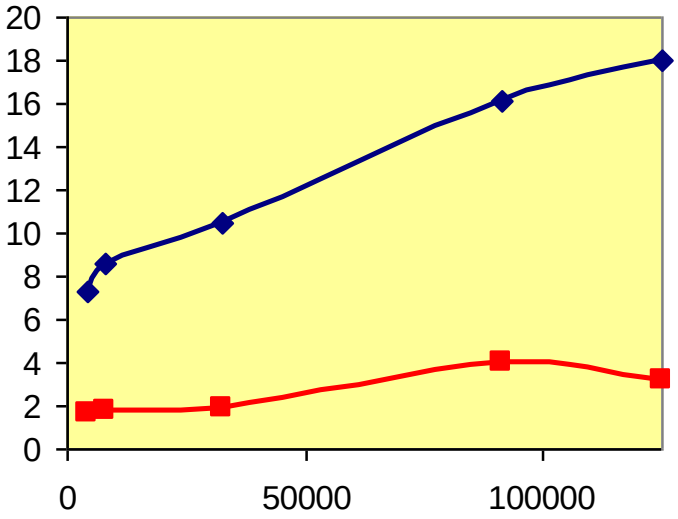


Runs performed on Thunder, Intel Itanium2 machine at LLNL
with 1024 nodes of 4 processors each (#2 on Top500 in June 2004)

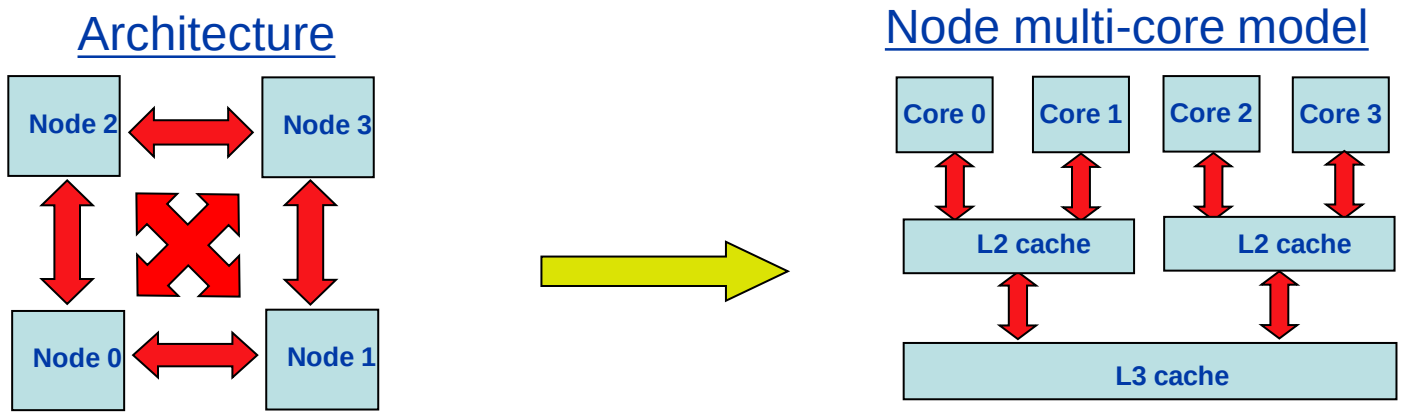
AMG-GMRES(10), approximately 90,000 degrees of freedom per process

New computer architectures with very large number of cores

- IBM BG/L appeared as #1 on the Top500 in November 2004, with 32,768 cores displacing Japan's Earth Simulator (5,120 cores, #1 from 2002 through mid 2004)
- Stayed there with increasing number of cores up to 212,992 in 2007
- AMG-CG achieved excellent scalability on BG/L using new algorithms up to 125,000 cores (25x25x25 dofs per core)
- Life is good!



Advances in computer architectures lead to new challenges!

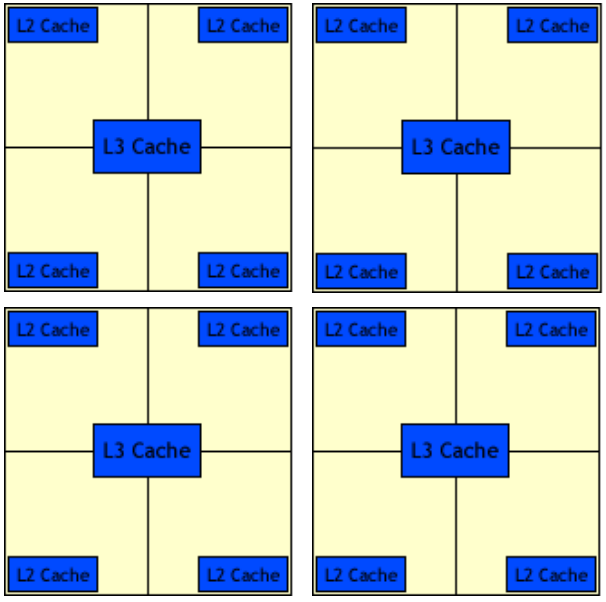


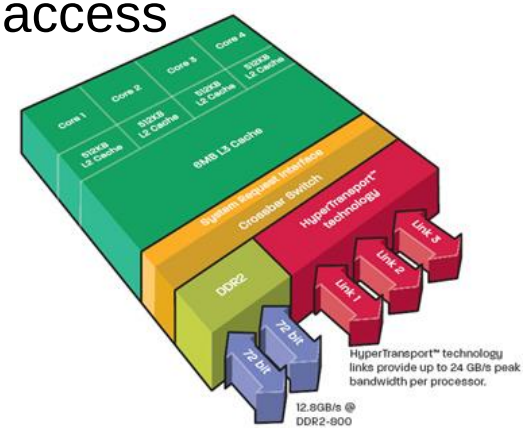
New architectures have

- multiple higher performing cores per node
- deeper memory hierarchies
- different networks

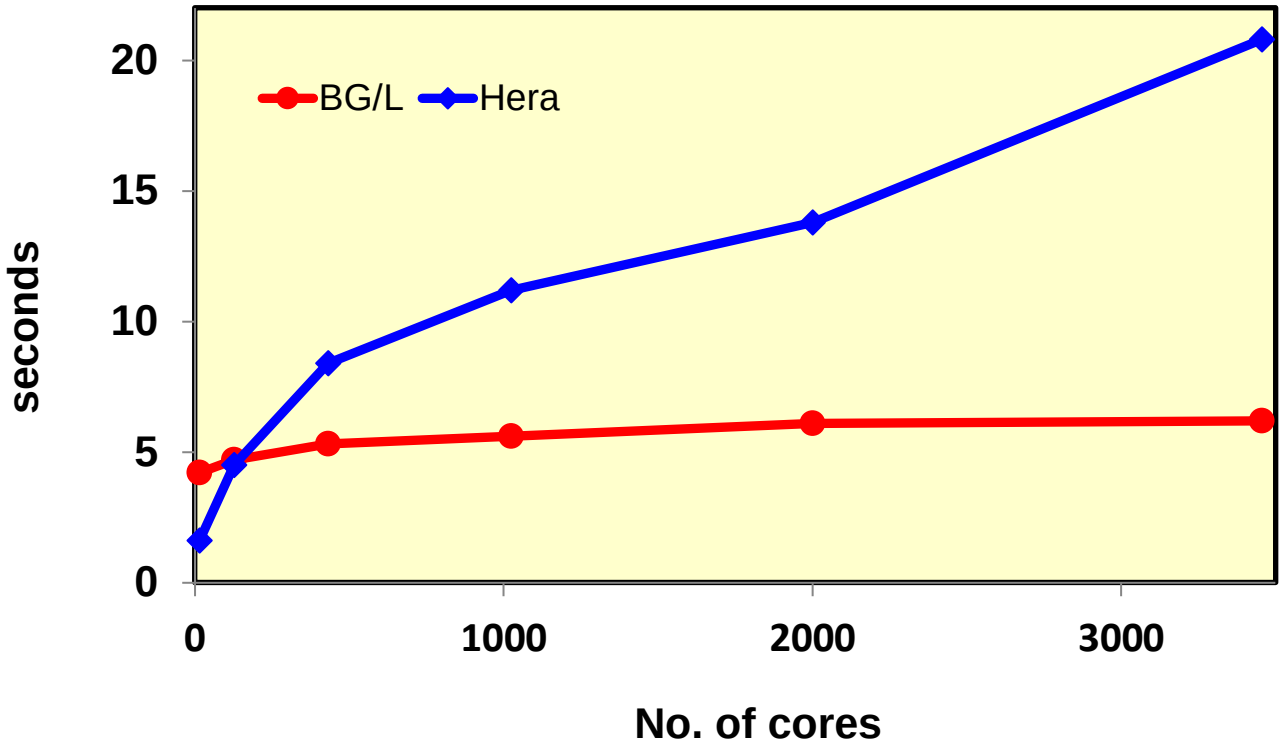
Multicore cluster details (Hera):

- Hera (Cray/HPE) 13,552 cores
- #32 Top500, June 2009
- 4 sockets per node, equipped with AMD Quad-core processors, connected by QDR Infiniband.
- Individual 512 KB L2 cache for each core
- 2 MB L3 cache shared by 4 cores
- 16 cores sharing the 32 GB main memory
- NUMA memory access



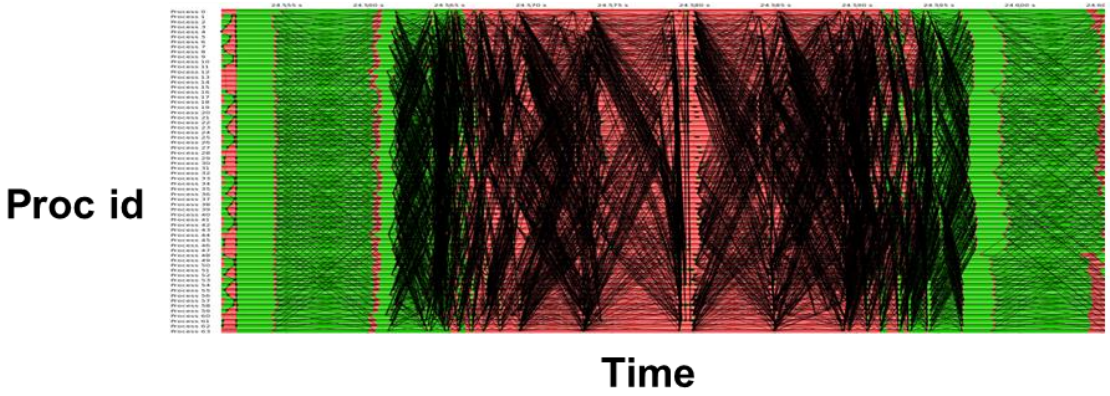


But …



While AMG scales almost perfectly on BG-type architectures, its weak scalability on a multi-core architecture is severely degraded.

What is causing the performance degradation?



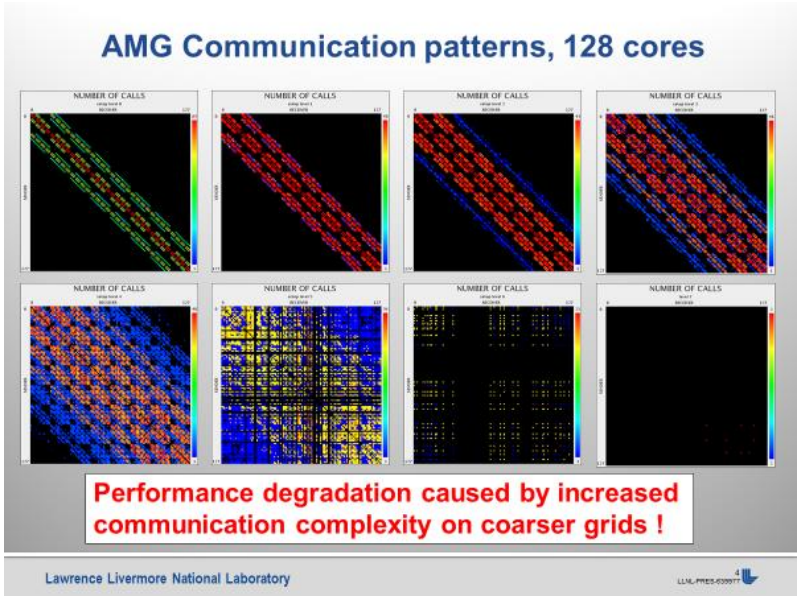
Performance profile of AMG solve cycle for 64 MPI tasks on Hera

computation

idle time

MPI calls

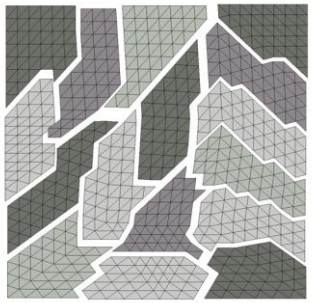
- AMG setup communication
- Coarse-grid selection in AMG can produce unwanted side effects
- Operator (RAP) “stencil growth” reduces efficiency

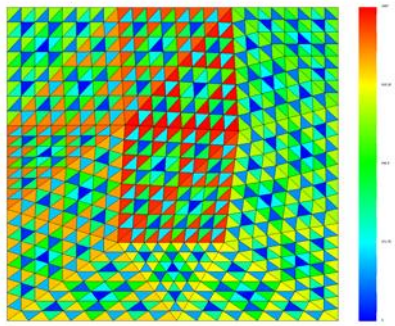


Efforts to reduce communications

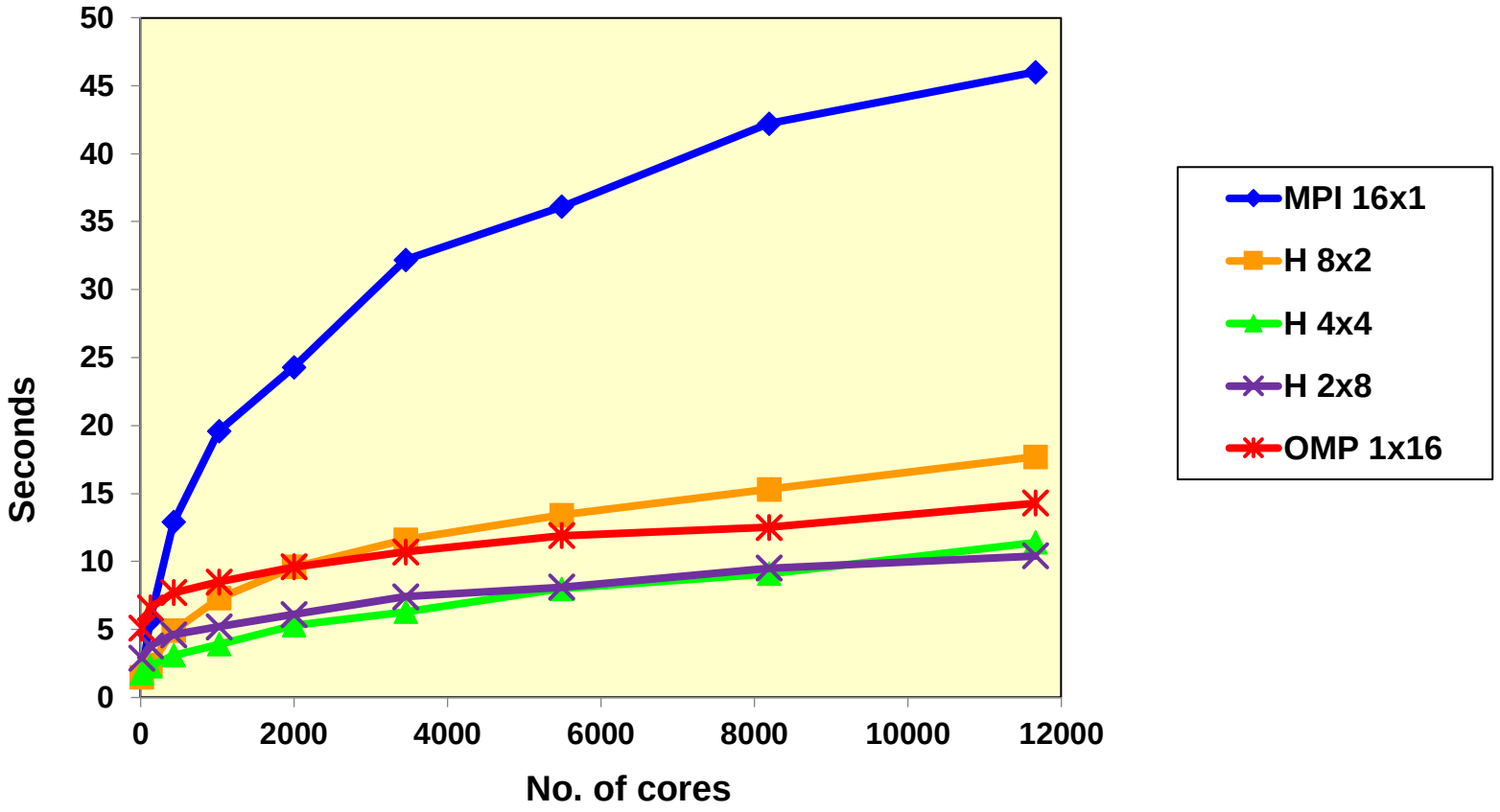
- Addition of a new programming model on node: OpenMP
- Now have hybrid programming model: MPI+OpenMP
- Requires adding OpenMP; straight forward in many routines
- Revisiting hybrid Gauß-Seidel:
  Convergence can be degraded by:
  - increasing number of blocks
  - decreasing block sizes
  - use of threads (can lead to poor partitioning)

$$M_H = \begin{pmatrix} \blacktriangle\blacktriangledown & & \\ & \ddots & \\ & & \blacktriangledown \end{pmatrix}$$



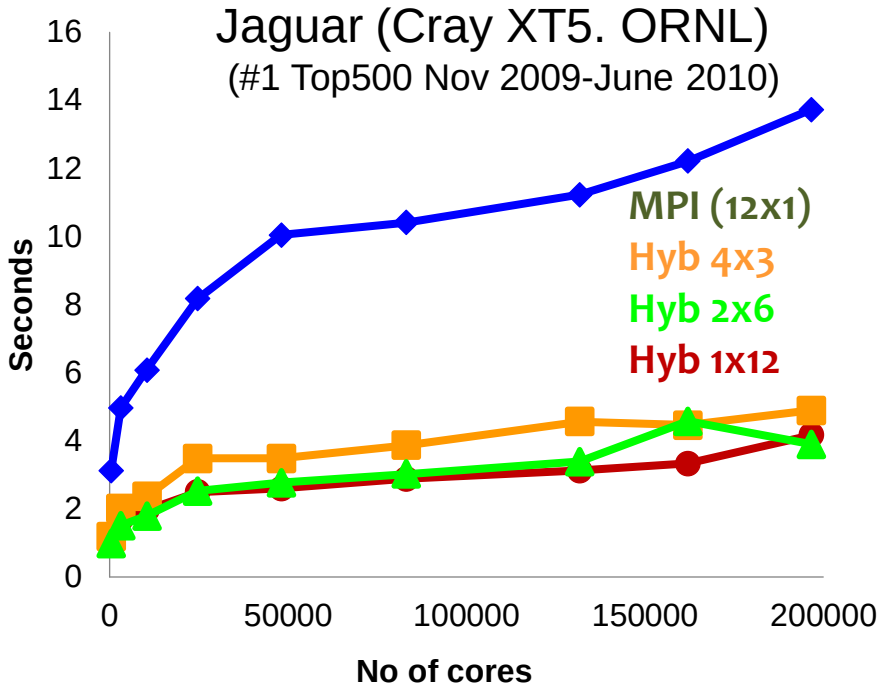


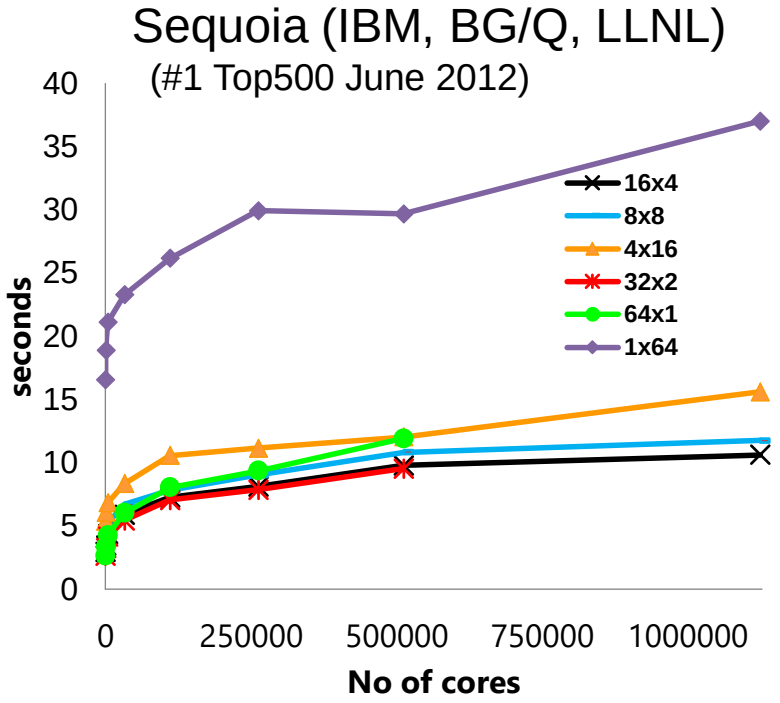
AMG-GMRES(10) on Hera, 7pt 3D Laplace problem on a unit cube, 100x100x100 grid points per node (16 procs per node)



We investigated AMG-Krylov solver performance with a hybrid OpenMP/MPI programming model on leading HPC platforms.

7pt 3D Laplace problem on a cube, weak scaling





- A general solution for obtaining good performance is not possible without considering the specific target architecture.

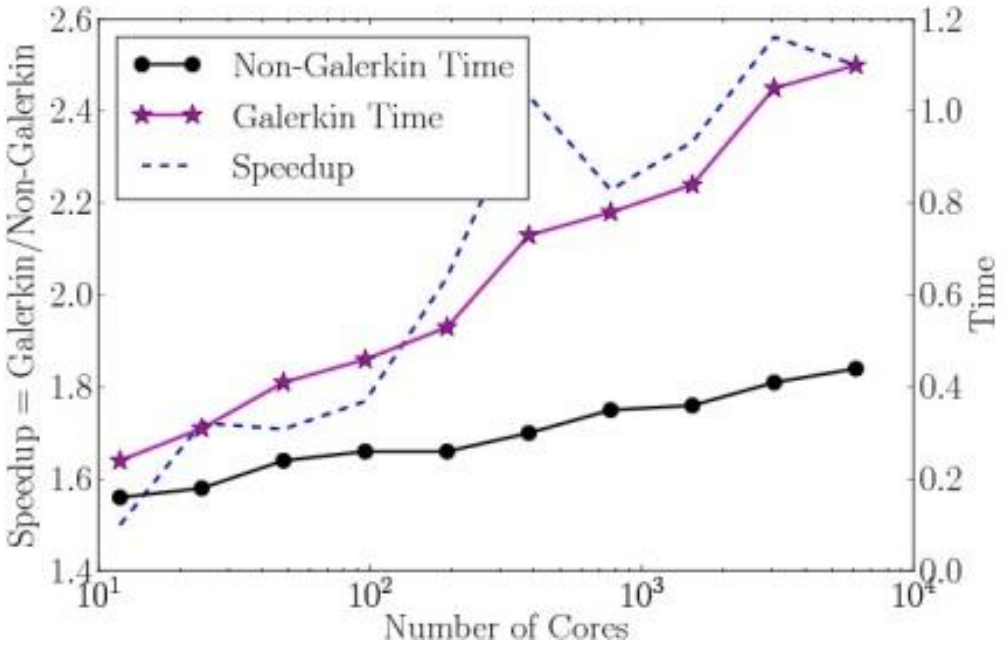
Baker, Gamblin, Schulz, UMY, Proceedings of IPDPS 2011.

Additional Efforts to Reduce Communication

- Agglomeration of coarse levels
- Reduction of number of nonzeros per row in Galerkin product
  - Use of interpolation truncation (was included in results shown earlier)
  - Sparsification of the coarse grid operator
- AMG-DD

Reducing parallel communication costs (1)

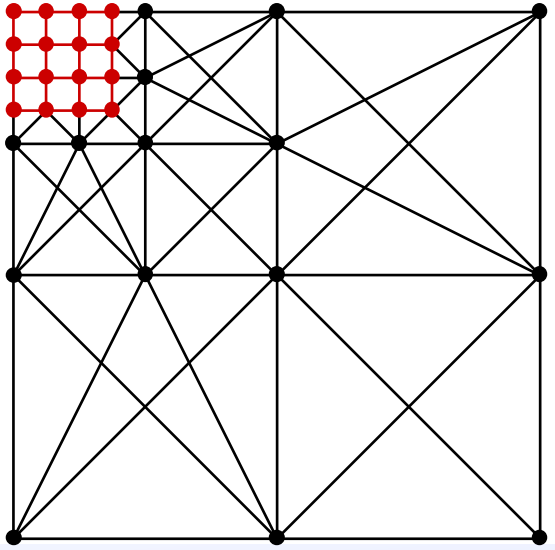
- Non-Galerkin AMG replaces the usual coarse-grid operators with sparser ones
  - Speedups from 1.2x - 2.4x over existing AMG
  - In *hypre* 2.10.0b



Falgout, Schroder, SISC, 36 (2014), pp. C309-C334.

Reducing parallel communication costs (2)

- 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 (vs $\log^2 N$)!
  - Implemented parallel code



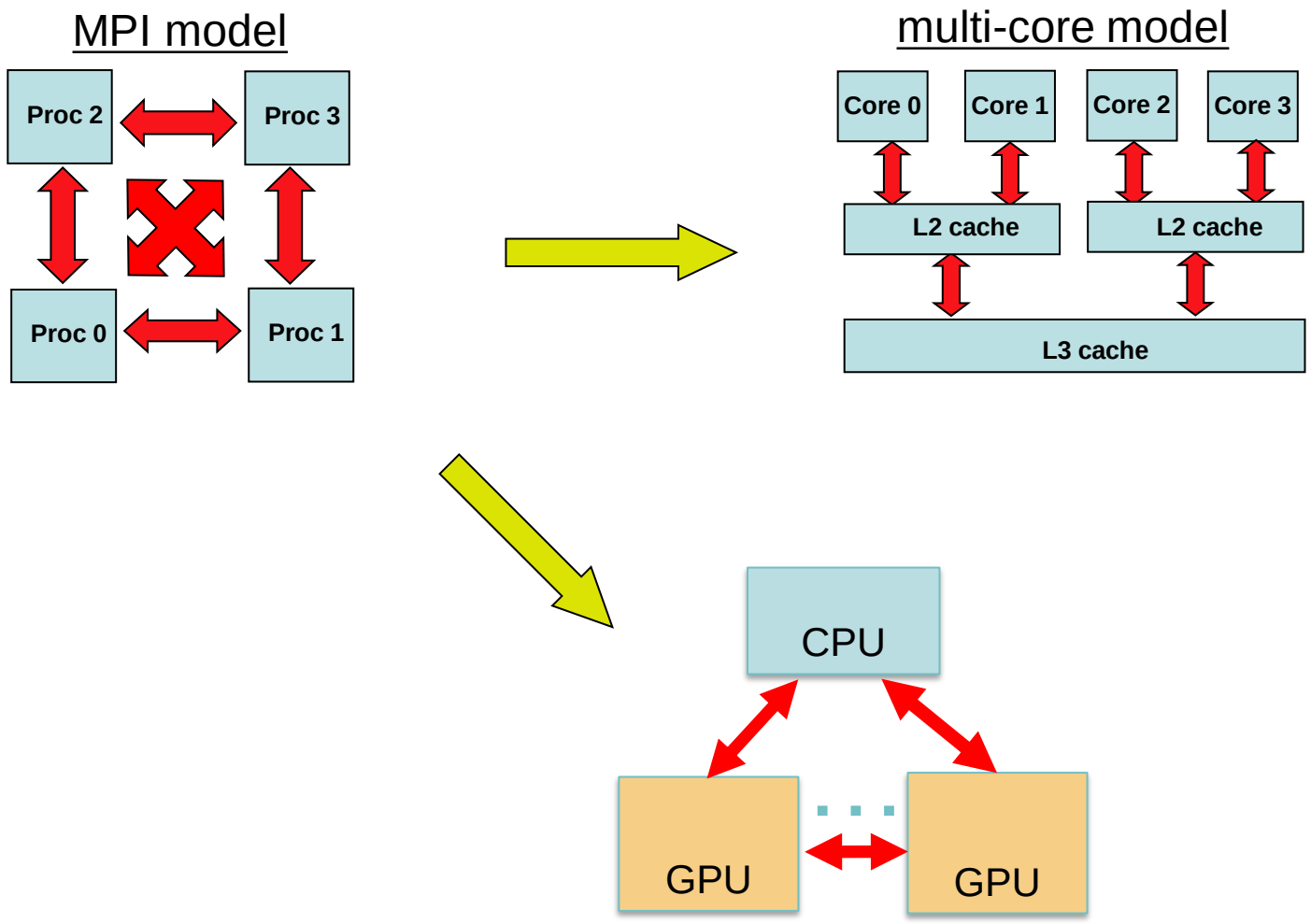
Mitchell, Strzodka, Falgout, NLAA, October 2020

Extreme Scale Challenges

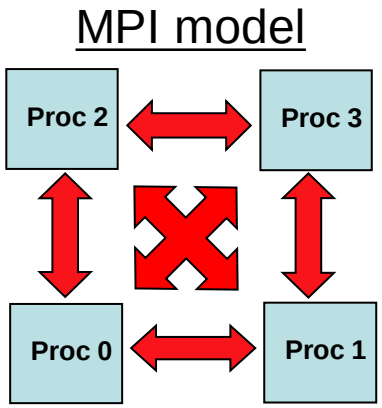
- Architectural Challenges at the Extreme Scale
  - Non-increasing clock speeds
  - Increasing number of cores
  - Limited power resources
  - Reduced memory per core
  - Heterogeneous architectures
  - Higher level of hardware failures

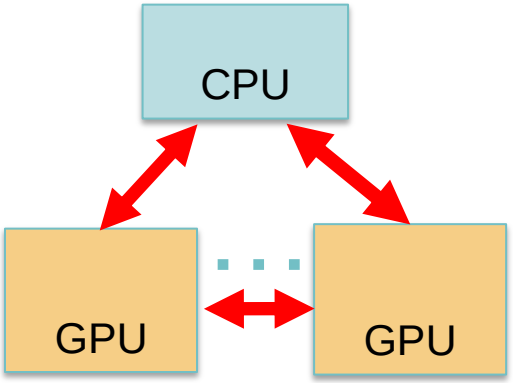
- Extreme Scale Solvers need to
  - Exhibit extreme levels of parallelism
  - Minimize data movement
  - Demonstrate resilience to faults
  - Be power efficient

Exascale Computers: Originally two pipelines



Exascale Computers: One was dropped!





Planned exascale computers

- Aurora, ANL, 2021
- Frontier, ORNL, 2021
- El Capitan, LLNL, 2022/2023

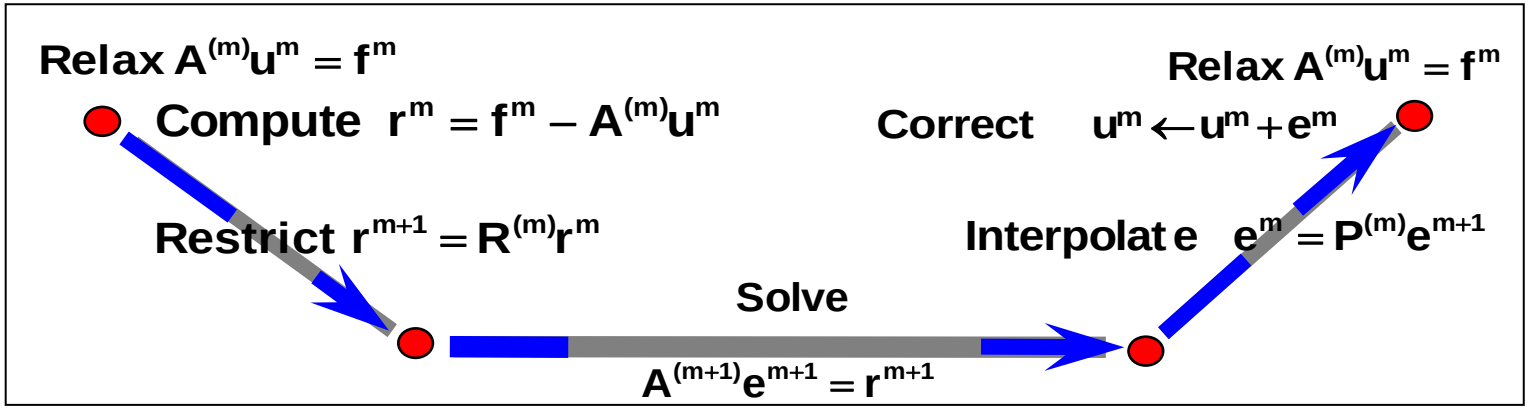
Strategy for unstructured interfaces/solvers in hypre in preparation for exascale platforms

- Modularize into smaller chunks/kernels to be ported to CUDA for Nvidia GPUS initially
- Develop new algorithms for portions not suitable for GPUs (interpolation operators, smoothers)
- Convert CUDA kernels to HIP for AMD GPUs and DPC++ for Intel GPUs (in progress)
- Various special solvers (e.g., Maxwell solver AMS, ADS, pAIR, MGR) built on BoomerAMG and will benefit from this strategy

Comments on GPU implementation

- Jacobi, polynomial smoothers are all based on matrix-vector multiplications
- MG solve phase can now completely be expressed in terms of matrix-vector multiplications
- Can take advantage of efficient GPU kernels!

Solve Phase:



Comments on GPU implementation of AMG Setup Phase

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

- Implement generation of strength matrix (highly parallel)
- Implement fine-grained parallel coarsening algorithm: PMIS (highly parallel)
- Implement coarse-grid operator generation – triple matrix product – much research for efficient matrix-matrix multiplication
- What about interpolation?

Interpolation operators suitable for GPUs

- While extended and extended+i interpolation are parallel and can be efficiently implemented on parallel multicore machines suitable for larger grain parallelism, not suitable for GPUs
- Due to memory-efficient implementation and need to distinguish fine and strong connections:
  many branches and if statements
  straight forward port unsuitable for GPUs
- Suitable interpolation: direct interpolation, but: generally, not well convergent, particularly at scale
- Solution: Design of a new class of interpolation operators that can be generated using basic matrix operations
  → class of MM-interpolation operators

How do you express interpolation in terms of multiplication between matrices?

MM-extended interpolation:

- Consider $A = \begin{bmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{bmatrix}$ and $A = A^s + A^w$, define $P = \begin{bmatrix} W \\ I \end{bmatrix}$
- Generate A^s_{FF} and A^s_{FC}
- $D_\beta = \text{diag}(A^s_{FC}1)$; row sums of A^s_{FC}
- $D_\alpha = \text{diag}\big(\text{diag}(A^s_{FF})\big)$; diagonal of A^s_{FF}
- $D_w = \text{diag}([A^w_{FF}, A^w_{FC}]1)$; row sums of weak coefficients
- Then

$$W = -(D_\alpha + D_w)^{-1}\big(A^s_{FF} - D_\alpha + D_\beta\big)D_\beta^{-1}A^s_{FC} = -\tilde{A}^s_{FF}\tilde{A}^s_{FC}$$

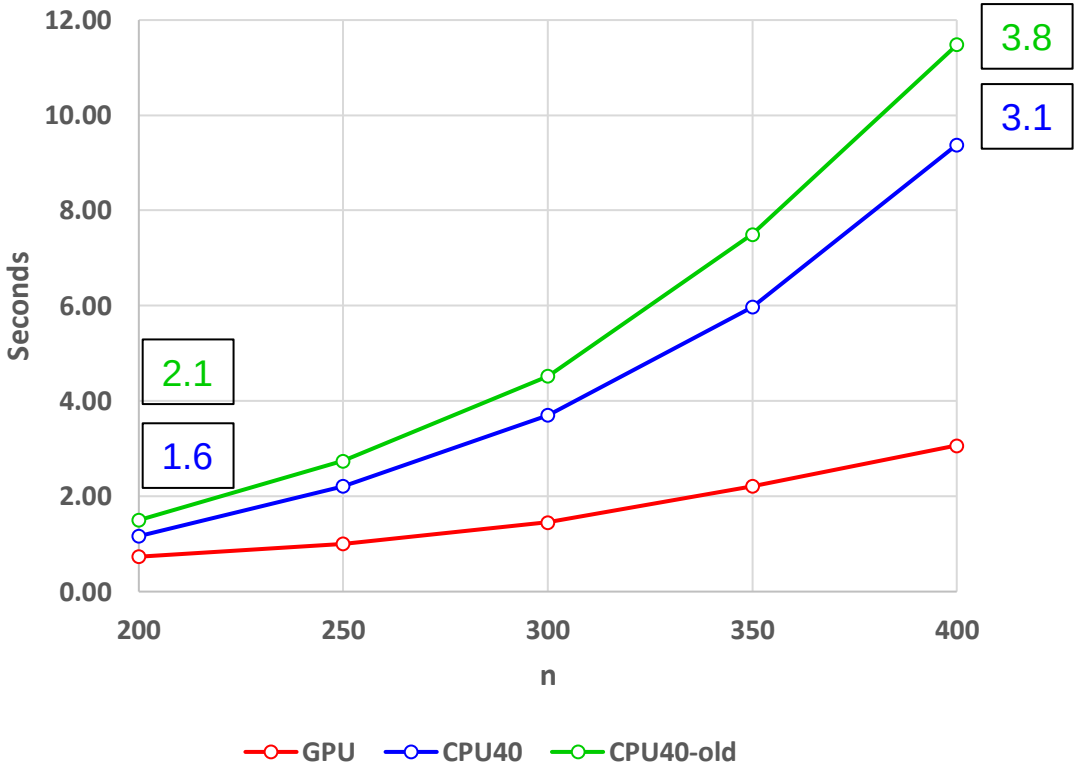
- Similar formulations for MM-extended+i interpolation and MM-extended+e interpolation

Li, Sjogreen, UMY, SISC, to appear.

Comparison CPU/GPU results on 16 nodes of Lassen (64 Nvidia GPUs, 640 Power 9 cores)

- Coupled 3D Poisson problem with 3 variables on a grid of size n x n x n, system size: $3n^3$, 375,000 to 3M dofs per GPU

- GPU, CPU40: MM-ext+i
- CPU40-old: ext+i

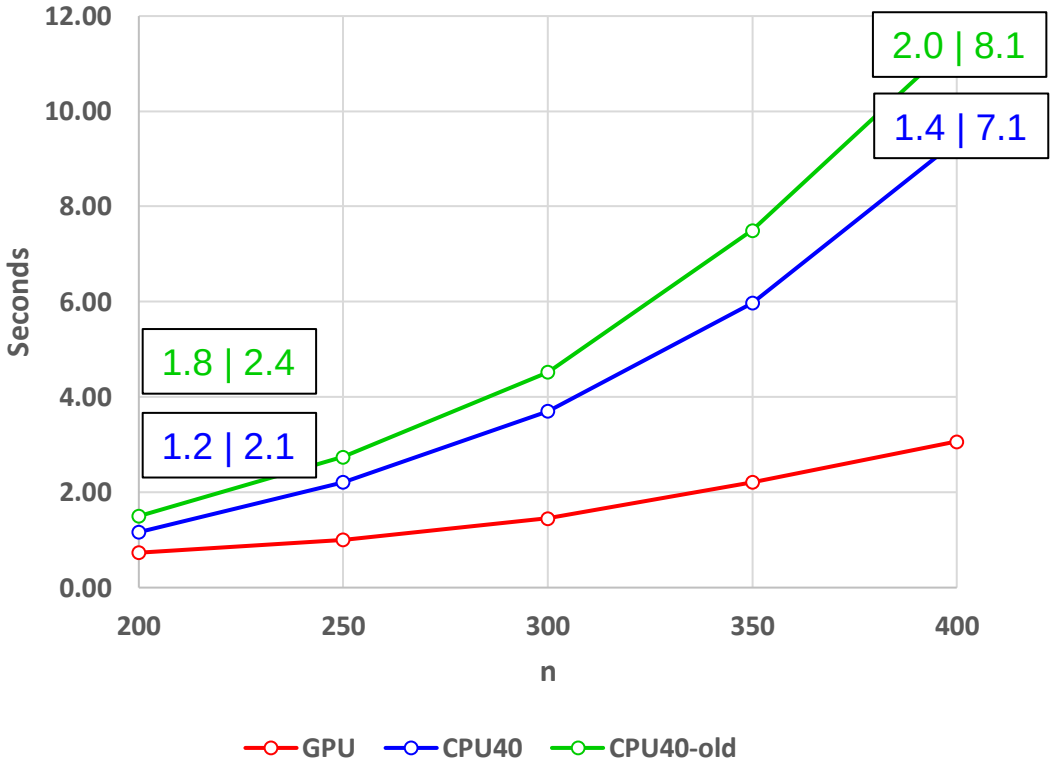


Comparison CPU/GPU results on 16 nodes of Lassen (64 Nvidia GPUs, 640 Power 9 cores)

- Coupled 3D Poisson problem with 3 variables on a grid of size n x n x n, system size: $3n^3$, 375,000 to 3M dofs per GPU

- GPU, CPU40: MM-ext+i
- CPU40-old: ext+i

Speedup: Setup | Solve



Conclusions/Future Work

- Hypre's AMG has gone through many changes as computer architectures have changed
- This required
  - New mathematical algorithms
  - Inclusion of new programming models
- Efforts to move AMG to different GPUs (AMD, Intel)
- Use of mixed precision in hypre
- New developments in hypre's structured/semi-structured interfaces
  - Development of a new semi-structured AMG algorithm in progress

Hypre developers and alumni

- **R. Falgout**
- A. Baker
- C. Baldwin
- A. Barker
- G. Castilla
- E. Chow
- A. Cleary
- N. Elliott
- H. Gahvari
- V. Henson
- E. Hill
- D. Hysom
- J. Jones
- T. Kolev
- M. Lambert
- B. Lee
- **R. Li**
- **S. Osborn**
- **D. Osei-Kuffuor**
- J. Painter
- **V. Paludetto Magri**
- J. Schroder
- B. Sjogreen
- C. Tong
- T. Treadway
- P. Vassilevski
- D. Walker
- L. Wang
- **U. Yang**

https://github.com/hypre-space/hypre http://www.llnl.gov/casc/hypre

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

Thank you!

This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project, a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.