Advances in hypre's GPU Capabilities

Ulrike Meier Yang

July 2, 2021

LLNL-PRES-823995. 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

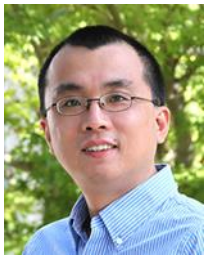
Lawrence Livermore National Laboratory

Acknowledgments

- Rob Falgout
- Ruipeng Li
- Victor Paludetto Magri
- Bjorn Sjogreen

- www.llnl.gov/casc/hypre

Rob Falgout



Ruipeng Li

Bjorn Sjogreen

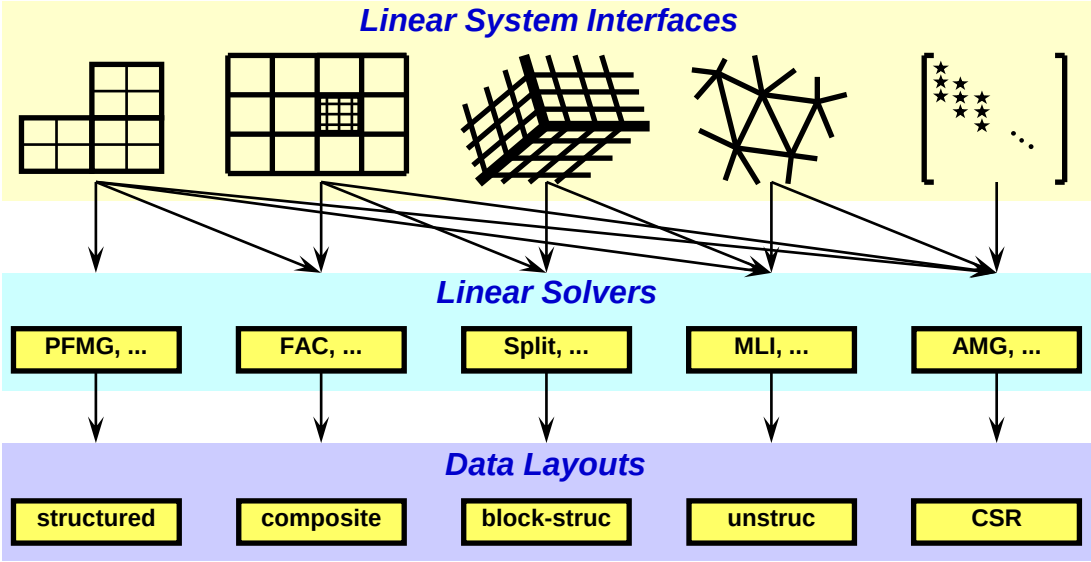
Victor Paludetto Magri

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 (for mixed-precision benchmark)
  - 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)

The need for redesign

- Current architecture trends favor regular compute patterns and large problems for high performance
- Difficult for unstructured solvers as well as for multigrid methods with decreasing level sizes
- Required different strategies in hypre's interfaces
- Hypre has 3 interfaces
  - Structured (Struct)
  - Semi-structured (SStruct)
  - Linear-algebraic (IJ)



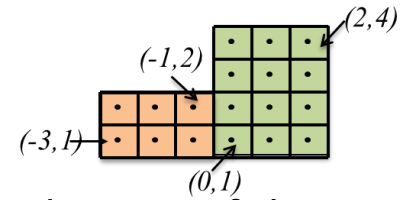
Strategy for Interfaces/Solvers in Preparation for Exascale Platforms

- Strategy for structured solvers:
  - More suitable data structures (based on grids and stencils) allow easier porting through adapting of the BoxLoops
    - Currently allow use of CUDA, **HIP**, RAJA, Kokkos, OpenMP 4.5

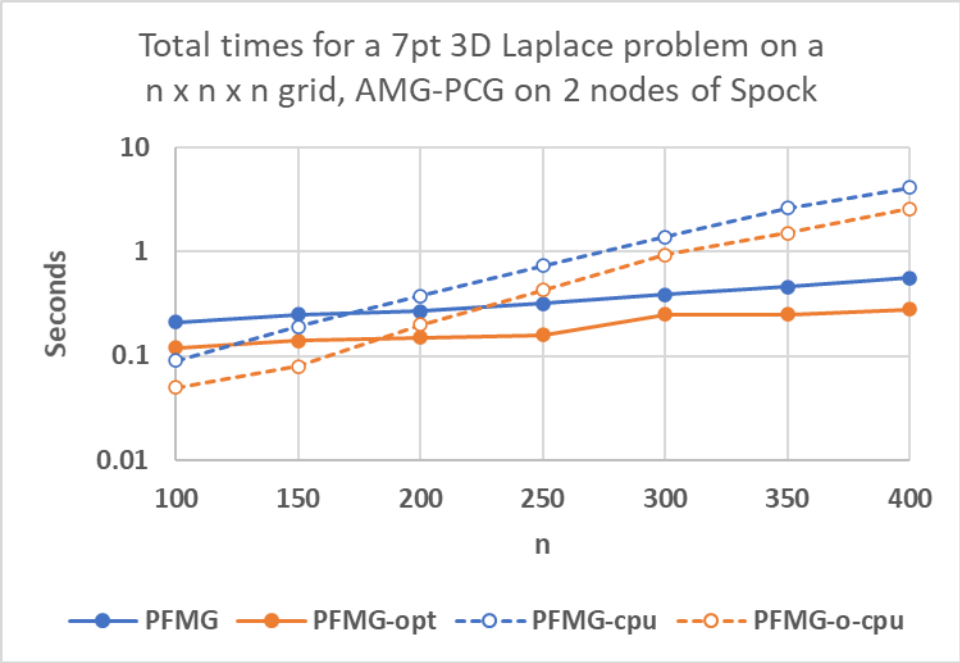
New

  - BoxLoops will also ease porting to Intel GPUs

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



- PFMG-PCG runs on Spock (4 AMD MI-100, AMD Rome per node)
  - CPU version: 4 MPI x 16 OMP per node
  - GPU version: 4 GPUs per node

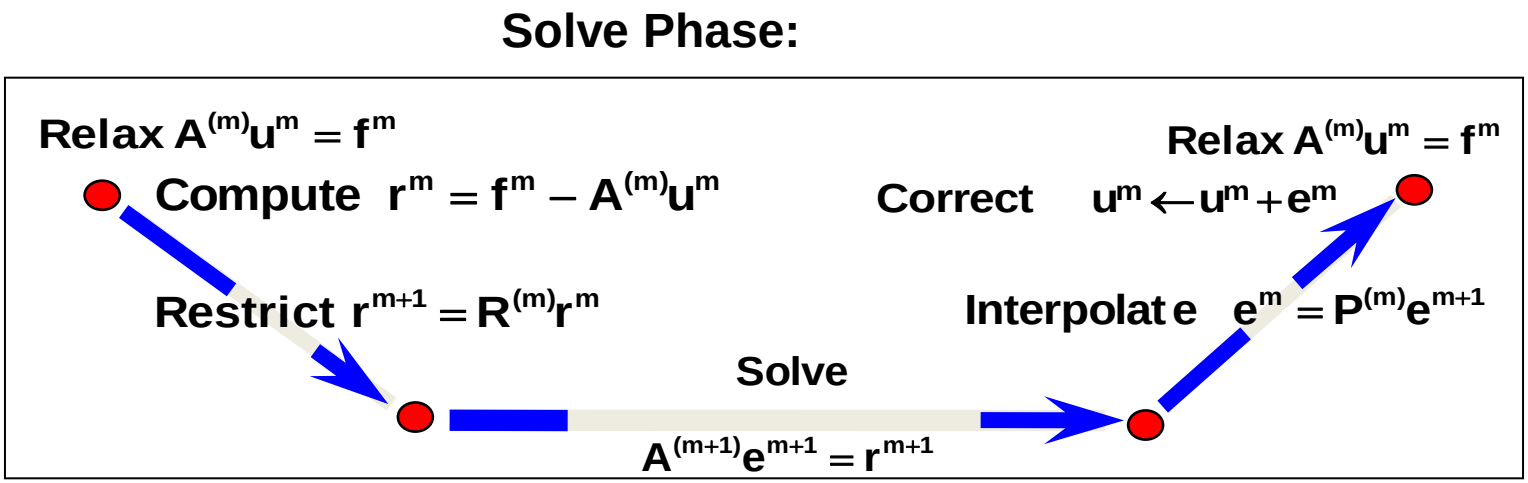


Strategy for Interfaces/Solvers in Preparation for Exascale Platforms

- Strategy for unstructured solvers, particularly for most often used AMG preconditioner/solver BoomerAMG
  - CSR-based matrix data structures
  - Modularize into smaller chunks/kernels to be ported to CUDA for Nvidia GPUS initially
  - Convert CUDA kernels to HIP for AMD GPUs (**partial support since hypre v.2.21.0!**) and SYCL for Intel GPUs
  - Develop new algorithms for portions not suitable for GPUs (interpolation operators)
  - Various special solvers (e.g., **Maxwell solver AMS, ADS, AME**, pAIR, MGR) built on BoomerAMG benefit from this strategy

Now CUDA-enabled!
New in v.2.22.0!

GPU Implementation of AMG Solve Phase



- 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!

GPU implementation of AMG Setup Phase

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

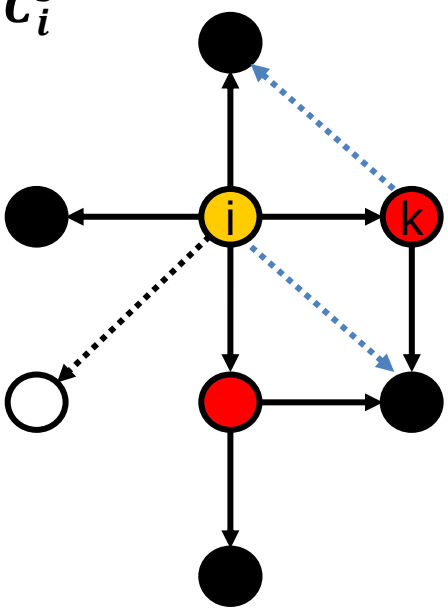
- Implement generation of strength matrix (highly parallel)
- Implement fine-grained parallel coarsening algorithm: PMIS (highly parallel)
- Implement coarse-grid operator generation – triple matrix product – much research for efficient matrix-matrix multiplication
- What about interpolation?

Distance-two interpolation operators

- Apply classical interpolation to extended stencil, $\hat{C}_i^s$
- Extended interpolation:

$$w_{ij} = -\frac{1}{a_{ii} + \sum_{k \in N_i^w - \hat{C}_i^s} a_{ik}} \left(a_{ij} + \sum_{k \in \boldsymbol{F_i^s}} \frac{a_{ik}\bar{a}_{kj}}{\sum_{m \in \hat{C}_i^s} \bar{a}_{km}} \right), \; j \in \boldsymbol{\hat{C}_i^s}$$

$$\bar{a}_{ij} = \begin{cases} a_{ij}, \text{ if } a_{ij}a_{ii} > 0 \\ 0, \text{otherwise} \end{cases}$$



- Weak connections to C-points complicate formula and implementation!

MM-Extended Interpolation

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

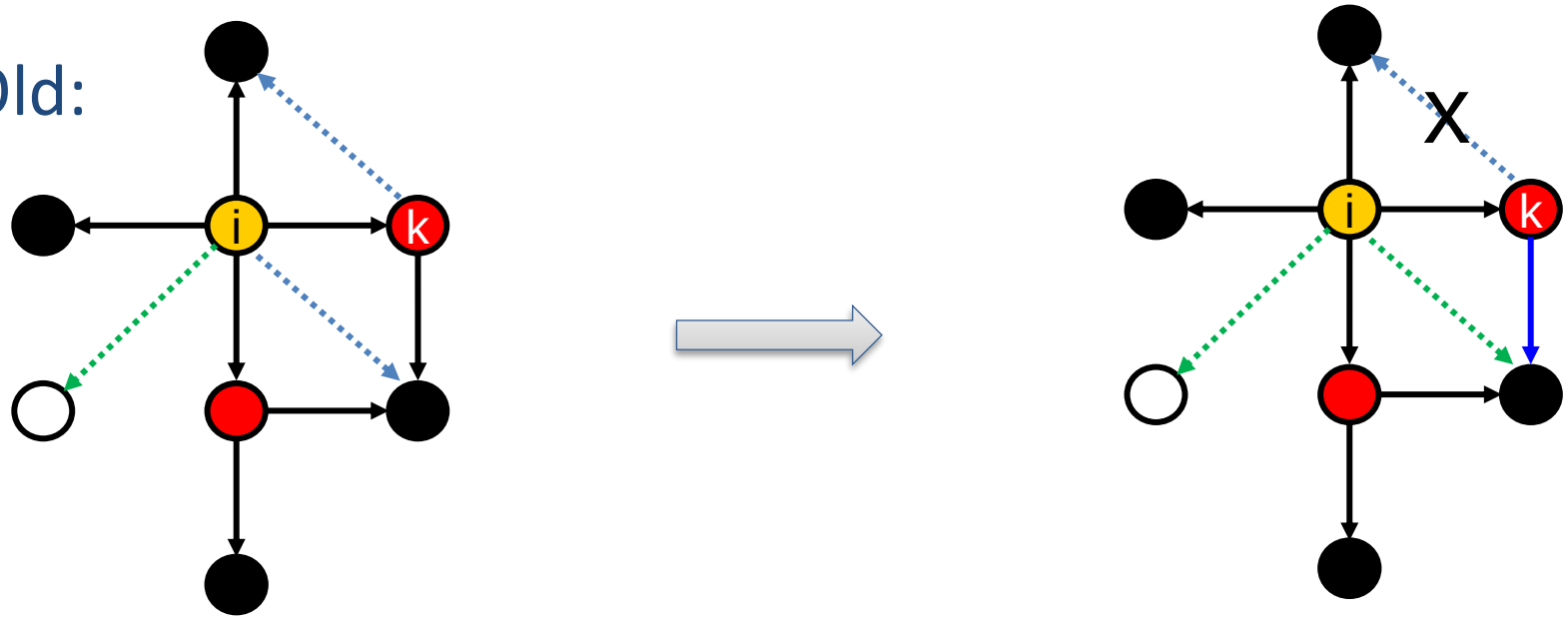
$$W = -(D_{FF} + D_w)^{-1}\left(A^s_{FF} + D_\beta\right)D_\beta^{-1}A^s_{FC} = -\tilde{A}^s_{FF}\tilde{A}^s_{FC}$$

Distance-two interpolation operators

- Extended MM-interpolation:

$$w_{ij} = -\frac{1}{a_{ii} + \sum_{k \in N_i^w} \boldsymbol{a_{ik}}} \left(a_{ij} + \sum_{k \in F_i^s} \frac{a_{ik} a_{kj}}{\sum_{m \in \boldsymbol{C_k^s}} a_{km}} \right), j \in \hat{C}_i^s$$

- Old:

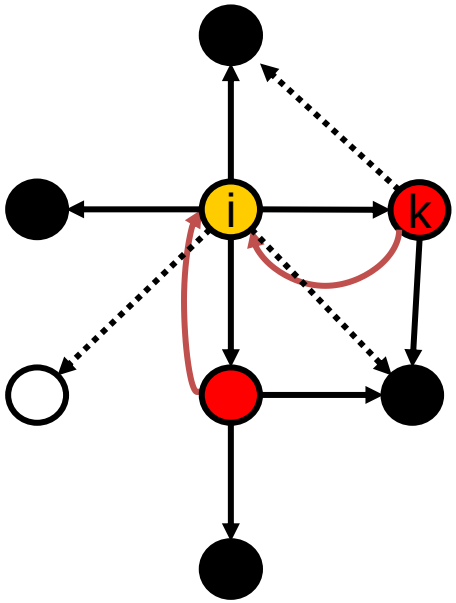


Distance-two interpolation operators

- Extended+i interpolation: include a_{ki}; add {i} to $\hat{C}_i^s$

$$w_{ij} = -\frac{1}{\tilde{a}_{ii} + \sum_{k \in N_i^w - \hat{C}_i^s} a_{ik}} \left(a_{ij} + \sum_{k \in F_i^s} \frac{a_{ik}\bar{a}_{kj}}{\sum_{m \in \hat{C}_i^s \cup \{i\}} \bar{a}_{km}} \right),$$

$$\tilde{a}_{ii} = a_{ii} + \sum_{k \in F_i^s} \frac{a_{ik}\bar{a}_{ki}}{\sum_{m \in \hat{C}_i^s \cup \{i\}} \bar{a}_{km}}$$



MM-Extended+i Interpolation

- Generate A^s_{FF} and A^s_{FC} (requires comm.)
- $D_{\beta,i} = \mathrm{diag}(A^s_{FF} e_i + A^s_{FC} 1)$; row sums of A^s_{FC} and i-th column of A^s_{FF}
- Generate $\hat{A}^s_{FF}$ by scaling rows of A^s_{FF} with $D^{-1}_{\beta,i}$ from right, i.e. i-th row of $\hat{A}^s_{FF}$ fulfills:

$$e_i^T \hat{A}^s_{FF} = e_i^T (A^s_{FF}) D^{-1}_{\beta,i} \quad \text{(requires comm.)}$$

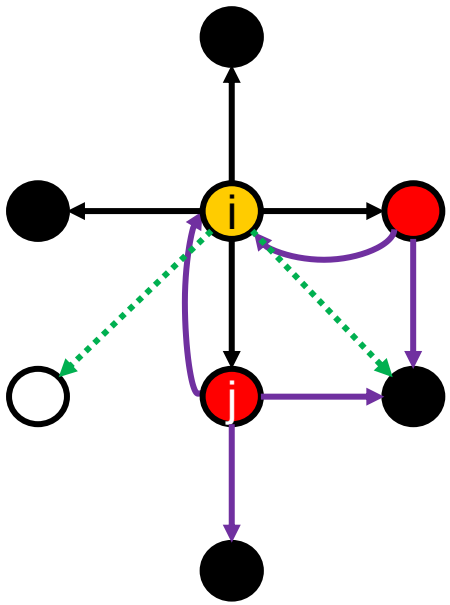
- $D_\theta = \mathrm{diag}\left(\mathrm{diag}\left(\hat{A}^s_{FF}(A^s_{FF})\right)\right)$; (only need diagonal of product)
- Then

$$W = -(D_{FF} + D_w + D_\theta)^{-1}\left(\hat{A}^s_{FF} + I\right) A^s_{FC} = -\tilde{A}^s_{FF} A^s_{FC}$$

- Requires more communication than extended interpolation!

MM-Extended+e Interpolation

- To avoid communicating rows and generating $D_{\beta,i}^{-1}$, we use an estimate for the connection to i:



$$a_{ji} \approx \sum_{m \in F_j^s} \frac{\boldsymbol{a_{jm}}}{\left|F_j^s\right|} =: \mu_j$$

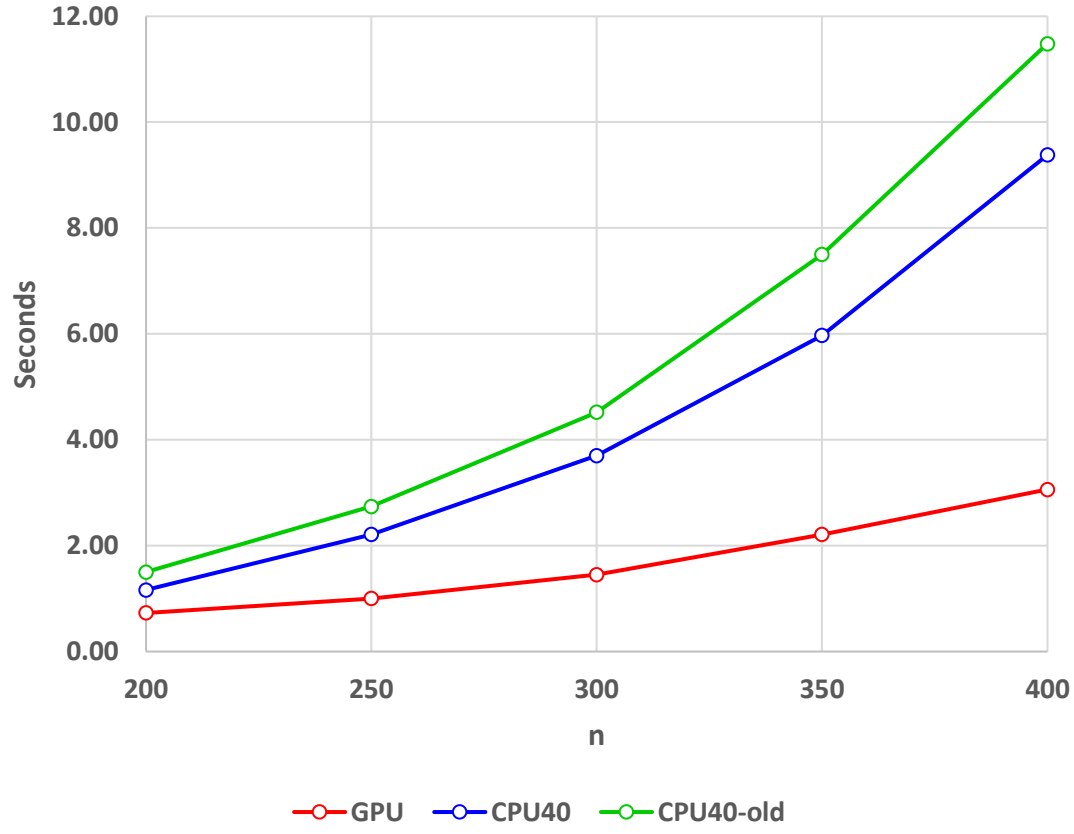
$$W = -(D_{FF} + D_w + D_\tau)^{-1}(A_{FF}^s + D_\lambda)D_\lambda^{-1}A_{FC}^s = -\tilde{A}_{FF}^s\tilde{A}_{FC}^s$$

with $[D_\lambda]_{ii} = [D_\beta]_{ii} + \mu_i$; $[D_\tau]_{ii} = \sum_{j \in F_i^s} a_{ij}\mu_j[D_\lambda]_{jj}^{-1}$

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



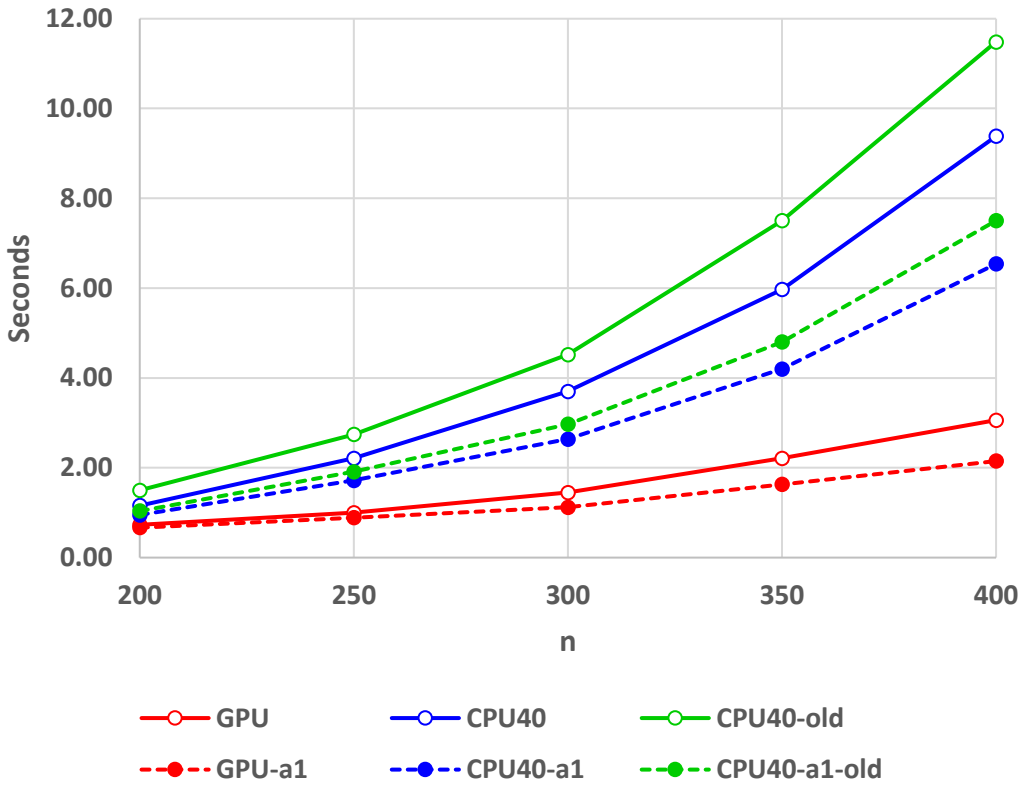
Comparison CPU/GPU results on 16 nodes of Lassen (64 Nvidia GPUs, 640 Power 9 cores)

- 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

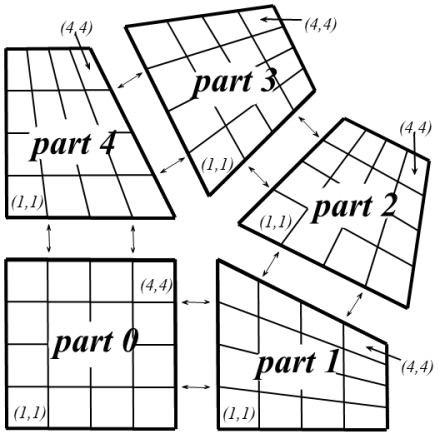
- a1: 1 level of aggr. coars.
- CPU40-a1-old: multipass

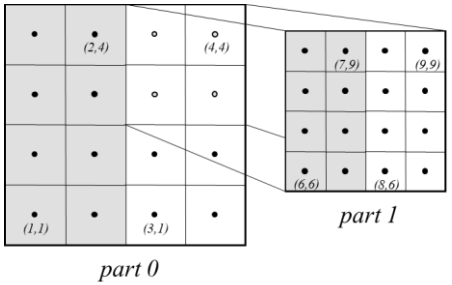
- GPU-a1, CPU40-a1: 2-stage MM-ext+e



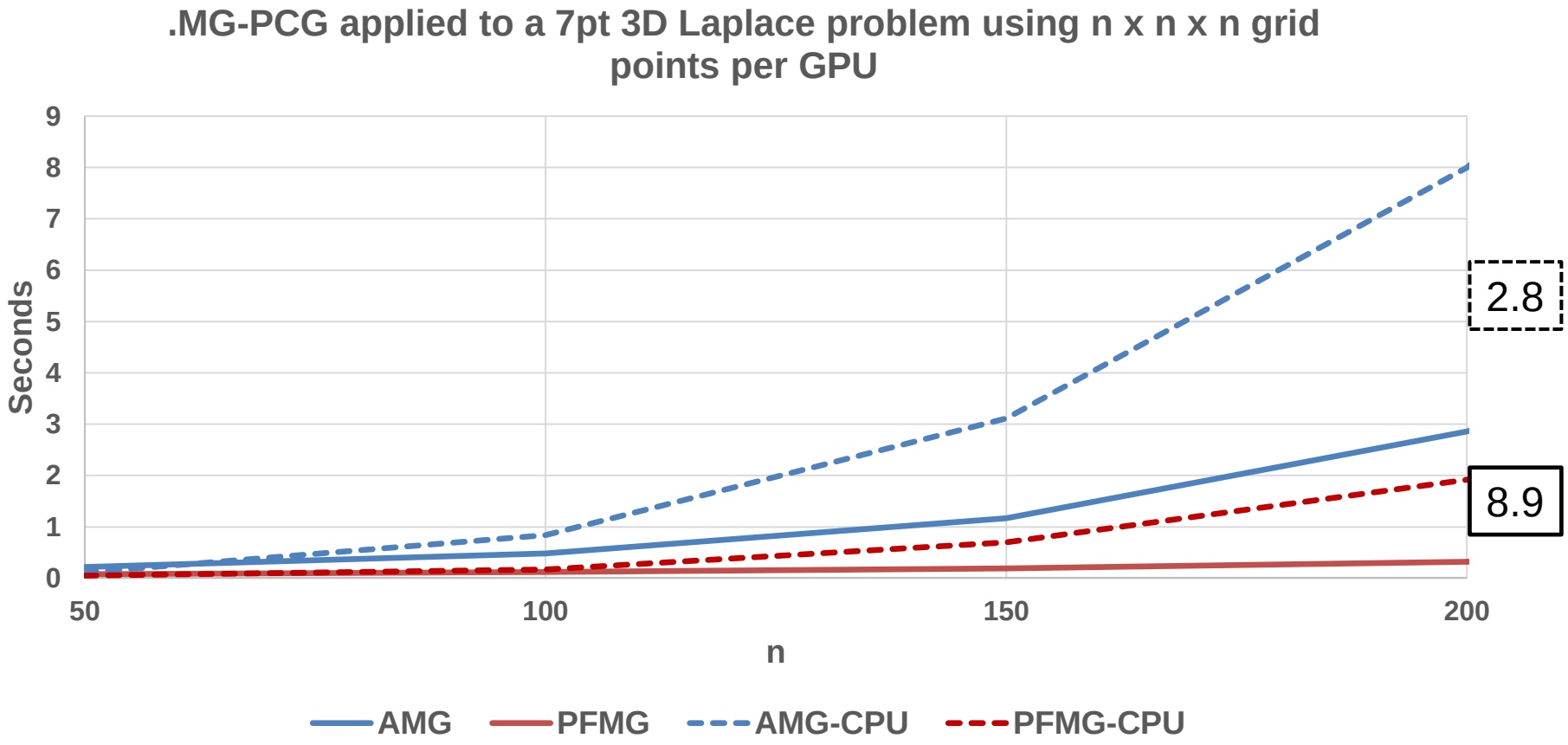
Strategy for interfaces/solvers in preparation for exascale platforms

- Strategy for semi-structured interfaces/solvers:
  - Mostly built on structured interface with some unstructured parts, e.g., semi-structured matrix A = S+U
  - Contains more suitable data structures than IJ interface due to structured portion, but able to express more complex problems than structured interface
  - Redesign of Struct/SStruct interface in progress that allows rectangular structured matrices
  - Semi-structured AMG solver in development

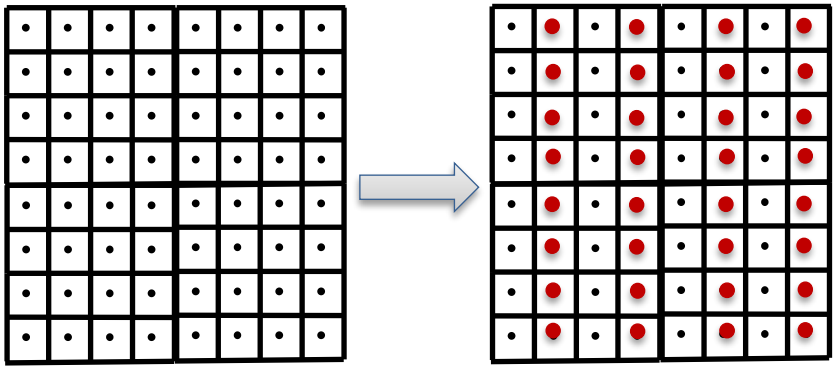




Structured multigrid methods perform significantly better than unstructured ones



PFMG is an algebraic multigrid method for structured grids



- Matrix A defined in terms of grids and stencils
- Uses semi-coarsening
- Simple 2-pt interpolation P
  → limits stencil growth in coarse grid operator $P^T AP$ to at most 9pt (2D), 27pt (3D)
- Optional non-Galerkin approach (Ashby, Falgout), uses geometric knowledge, preserves stencil size
- Pointwise smoothing
- Highly efficient for suitable problems

Semi-Structured Algebraic Multigrid Solver

- SStructMatrix is split into structured and unstructured couplings: $A = S + U,$ where U is a very small portion of the matrix

- Generate interpolation and restriction for structured parts only:

$$P_S, \qquad R_S = P_S^T$$

  Adjust weights at part boundaries as needed

- Generate coarse grid operator:

$$A_C = R_S A P_S = R_S S P_S + R_S U P_S = S_C + U_C$$

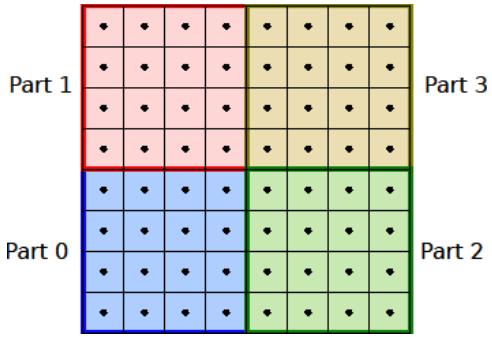
  mostly structured ops with good potential for efficient performance

- With simple 2-point interpolation, U_C continues to be restricted to part boundaries, no growth into interior

- Apply weighted Jacobi smoother to $S_C + U_C$

Software development of SSAMG

- Work in progress, built on redesigned Struct interface (not optimized yet)
- Implemented SSAMG method, but not fully optimized yet, i.e., view presented times with caution!
- Shows overall good convergence for Laplace type problems on block-structured grids, e.g.,



system sizes varying from 1M to 268M

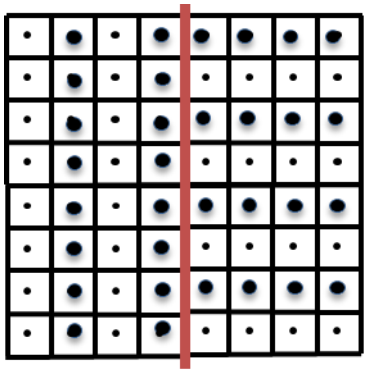
# procs	1	4	16	64	128
# iters	11	12	13	13	12
times	3.1	4.2	5.0	6.9	7.2

Results for 2 parts 3D mixed anisotropy (0.01) (MG-GMRES(10)) on a Linux box

	SSAMG	AMG-4pt	AMG-2pt	AMG-4pt 1 lvl agg. c.
Setup time	6.3	15.8	10.1	6.8
Solve time	12.9	12.7	18.0	13.0
iterations	11	12	21	22

Total grid size: 192 x 192 x 192
System size: 7,077,888

AMG: PMIS coarsening,
weighted Jacobi (0.7)

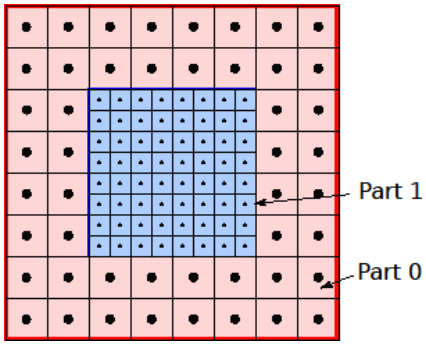


Results for 2 parts 3D AMR (MG-PCG) on a Linux box

	SSAMG	AMG-4pt	AMG-2pt	AMG-4pt 1 lvl agg. c.
Setup time	4.3	11.7	7.6	4.5
Solve time	7.7	4.5	7.0	4.7
iterations	13	10	19	17

Grid size: 64 x 64 x 64 per part
System size: 4,194,304

AMG: PMIS coarsening,
weighted Jacobi (0.7)



Summary/Future work

- Development of MM-interpolation operators enabled implementation on GPUs
- Modularization improves performance portability
- Now partial HIP support in AMG, struct interface/solvers
- Early SSAMG results show potential,
  but still require much work!
- Continue development of SSAMG
  - Fix bugs and address performance issues
  - Investigate inclusion of unstructured portion into interpolation: $P_S + P_U$
- Investigate hybrid SSAMG/AMG option

Thank you!

This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), Scientific Discovery through Advanced Computing (SciDAC) program, 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.

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory