Performance Analysis of Coarse Solvers for Algebraic Multigrid on Leading Multicore Architectures

Sherry Li, Alex Druinsky, Osni Marques, Pieter Ghysels, Sam Williams
Lawrence Berkeley National Laboratory

Andrew Barker, Panayot Vassilevski
Lawrence Livermore National Laboratory

Delyan Kalchev
University of Colorado, Boulder

14th Copper Mountain Conference, March 20 – 25, 2016

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

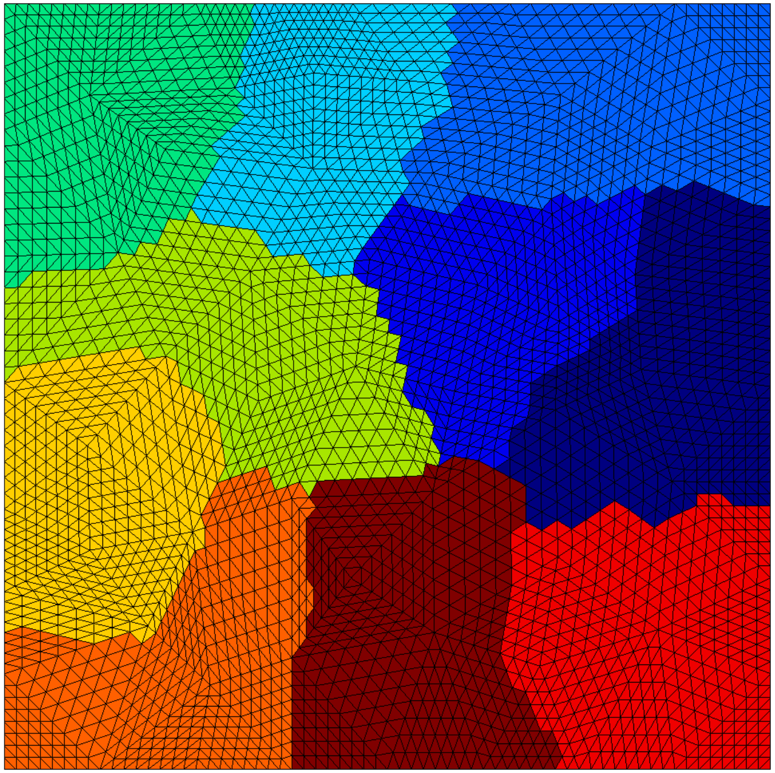
Outline

- **New spectral element AMG method**
  - Set up hierarchy
  - Coarse grid solver
- **Performance characterization, multicore optimization, bound analysis**
  - Intel MIC Knights Corner
- **Software components**

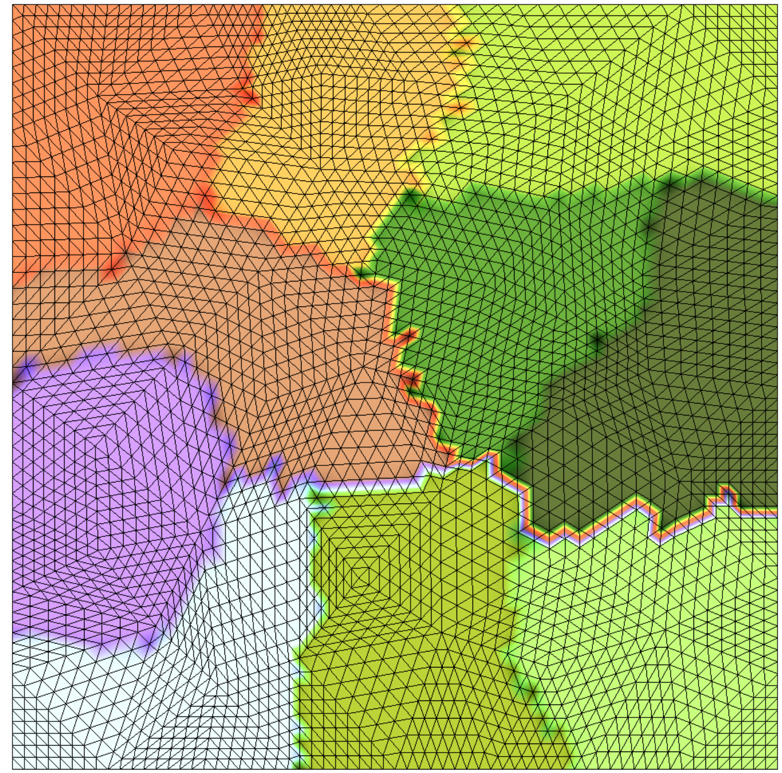
Advances in multigrid solver: Smoothed aggregation spectral element AMG

- **AMGe is geometric multigrid with nonstandard elements (agglomerates of fine-grid ones) and operator-dependent coarse finite element spaces**
- **For elliptic problems (diffusion and elasticity), solve local eigenvalue problems to build AMG hierarchy – highly parallel**
  - Code SAAMGe released
- **For more general problems (electromagnetics and Darcy flow), solve local SVD problems to build AMG hierarchy – highly parallel**
  - Prototype code ParElag
- **Novelty: The hierarchy can be employed for nonlinear solvers on unstructured meshes as well as for MCMC simulations**
  - In contrast to plain AMG, AMGe coarse spaces have guaranteed approximation properties so the coarse problems provide highly accurate discretizations (useful for nonlinear problems and for dimension reduction)

Agglomeration and Aggregation in finite elements



elements → agglomerates



vertices → aggregates

Smoothed aggregation spectral element AMG (SAAMGE)

- **High order elements to discretize PDEs**
- **Algebraic multigrid (AMG) solver**

1. Pre-smoothing (fine grid)
   - Intermediate iterate: $y = x_i + M^{-1}(b - A\,x_i)$
2. Coarse-grid correction (recursion → multilevel)
   1) Restrict the residual: $r_c = P^T(b - A\,y)$
   2) Solve coarse-grid defect equation: $A_c x_c = r_c$
   3) Interpolate, compute next intermediate iterate: $z = y + P\,x_c$
3. Post-smoothing (fine grid)
   - $x_{i+1} = z + M^{-T}(b - A\,z)$

PCG, hypre, HSS, . . .

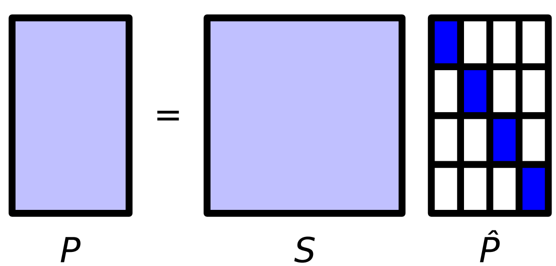
Interpolation matrix P for coarse space construction (set up AMG hierarchy)

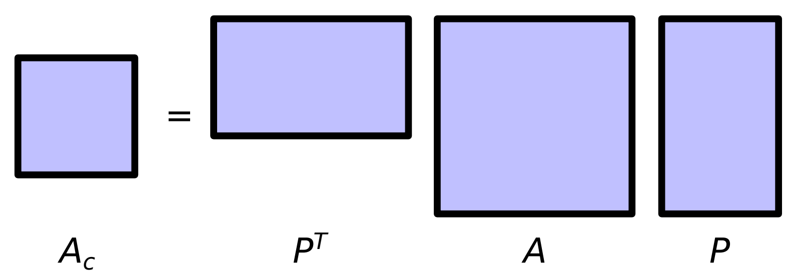
- **Coarse-grid matrix $A_c = P^T A P$; P is computed from lowest eigenvectors of local element matrices**
  - Improves approximation quality of coarse space, but expensive to compute
- **Tentative**

$$P = \begin{bmatrix} P_1 & & & \\ & P_2 & & \\ & & \ddots & \\ & & & P_{na} \end{bmatrix}$$

P_i is obtained by solving eigenvalue problem $A_i\, q = \lambda\, D_i\, q$, need several smallest eigenpairs

Final P = S P, where S is a matrix polynomial (e.g. Chebyshev)





Interpolation matrix P for coarse space construction

- **Two improvements:**
  - Replace dense LAPACK by sparse ARPACK + SuperLU
    - Projection method reduces to smaller problems
  - Developed an early termination scheme by monitoring the accuracy of already computed eigenpairs
    - Using implicit QL method for tridiagonal matrices

  → over 2x faster in total solution time

Osni Marques, Wednesday, 10:50 am,
"Tuning the Coarse Space Construction in a Spectral AMG Solver"

Advances in low-rank HSS factorization solver

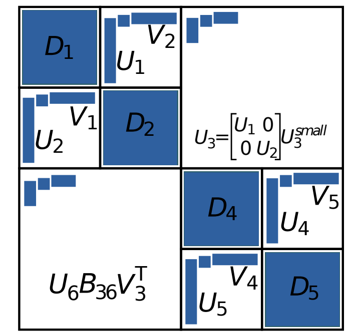
- **Goal: achieve O(N), or O(N polylog(N)) complexity**
  - Traditional sparse LU requires $O(N^2)$ Flops (e.g., SuperLU)
- **Approach: use hierarchical matrix algebra**
  - Accurate approximation with low-rank, data-sparse structures
- **Same mathematical foundation as Fast Multipole (FMM), but more general**
  - Diagonal block ("near field") exact; off-diagonal block ("far field") approximated via low-rank format

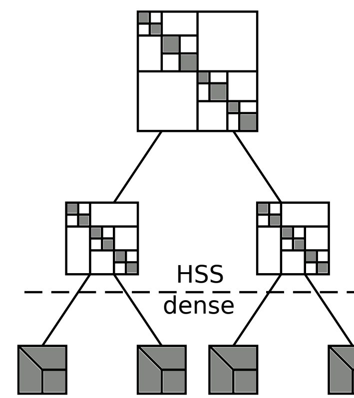
$$A \approx \begin{bmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{bmatrix}$$

- **Broad applications**
  - PDEs with smooth kernels, Integral equations, Boundary Element Methods, genreal-purpose preconditioners, …

Use hierarchical partitioning and nested bases to achieve lower complexity

- STRUMPACK, ~200 downloads in 2015
  - 5.4x faster than dense LU for BEM matrices
  - 7x faster than traditional sparse solver for PDEs, 4-fold memory reduction
    - CEMM fusion SciDAC problems (two-fluid MHD)
    - ComPASS accelerator SciDAC problems (Maxwell equations)



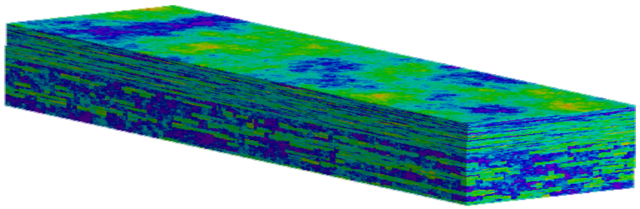


Pieter Ghysels, Wednesday , 5:45 pm

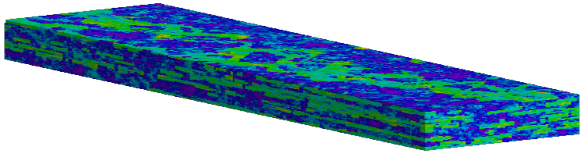
"Evaluation of a preconditioner using low-rank approximation and randomized sampling"

Benchmark: PDE with jump coefficients

- **SPE10 Benchmark (model 2) (www.spe.org/web/csp/)**
  - Formation in the Brent oil field; 1200'x2200'x170' (cell size 20'x10'x2').
  - Top 70 ft (35 layers) represents the Tarbert formation; bottom 100 ft (50 layers) represents Upper Ness (fluvial).



porosity of the whole model



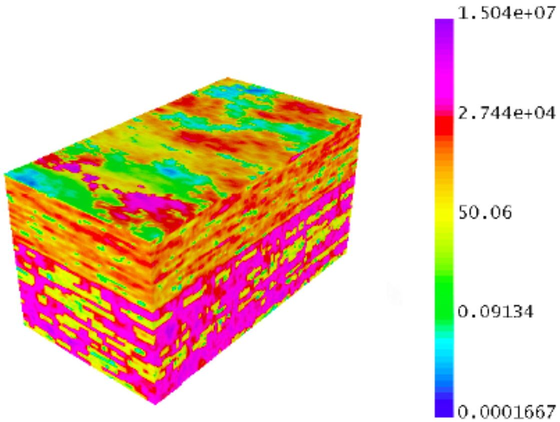
part of the Upper Ness sequence

- **Darcy equation**

$$-\nabla \cdot (k(x)\nabla p) = f(x), \forall x \in \Omega$$

$p(x)$ is pressure

$k(x)$ is permeability of the medium

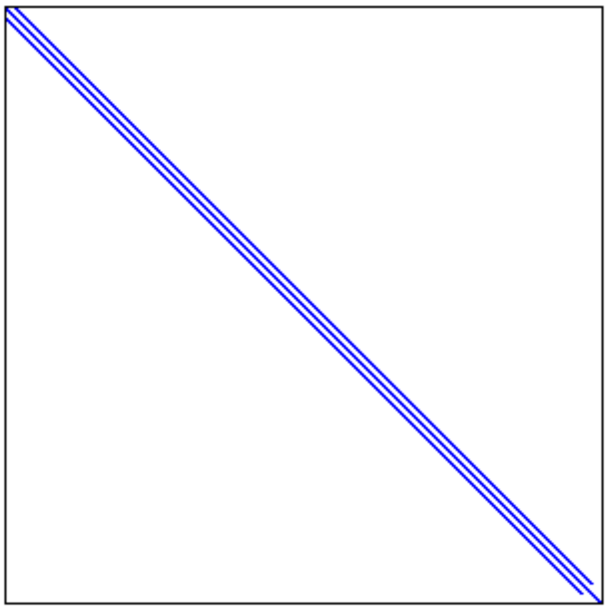


[Villa]

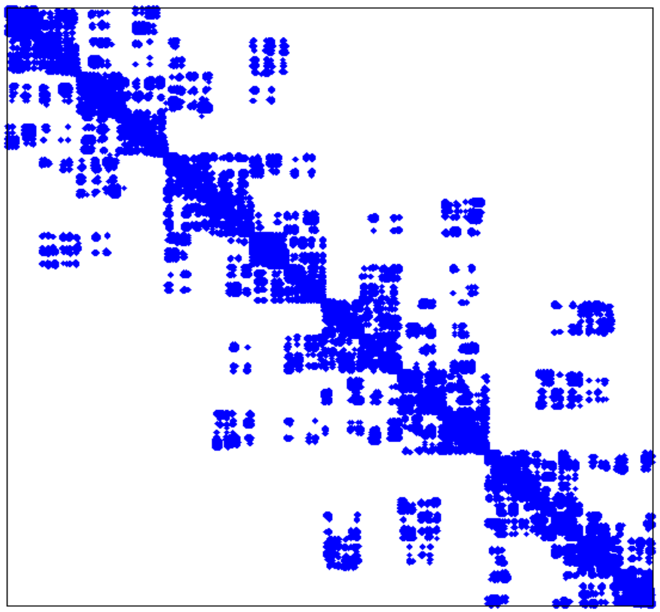
Two distinct soil layers and the large jumps in the coefficient k between them.

Fine-grid, coarse-grid matrices from SAAMGE

n ≈ 1.2 M; nnz ≈ 30.6 M; nnz/n ≈ 26



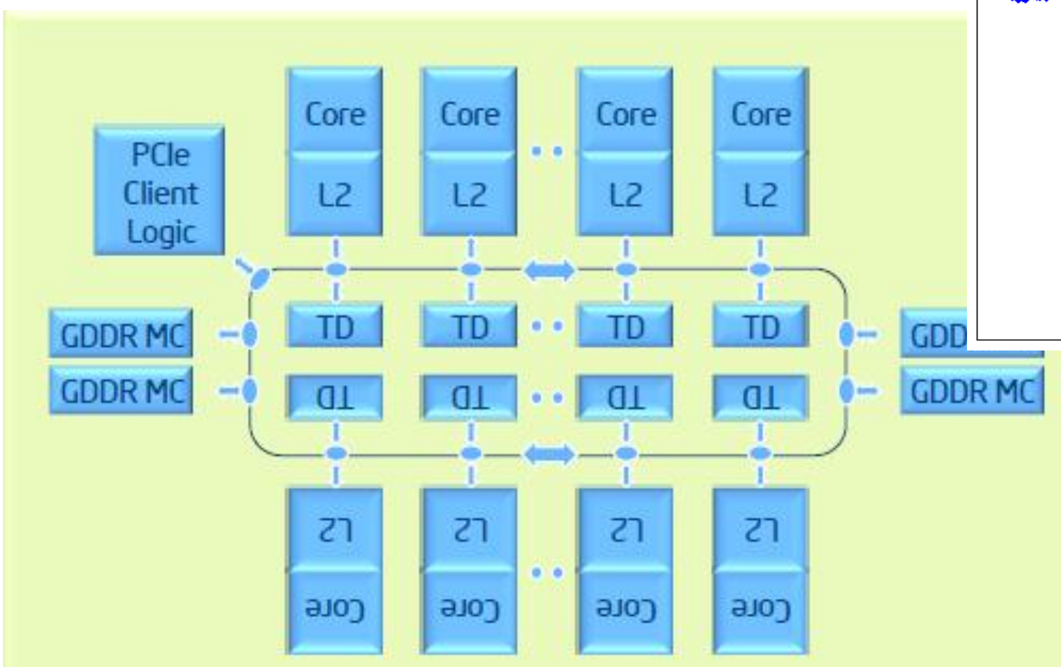
n = 7,782; nnz = 1,412,840; nnz/n = 181.6

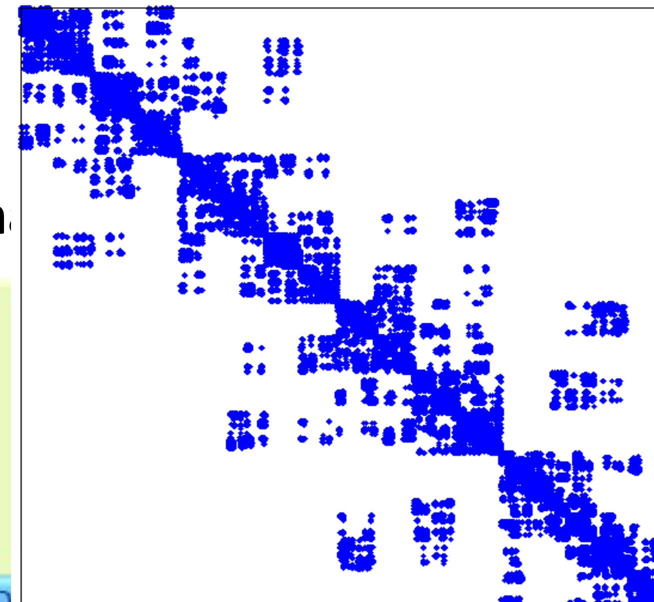


Multicore example

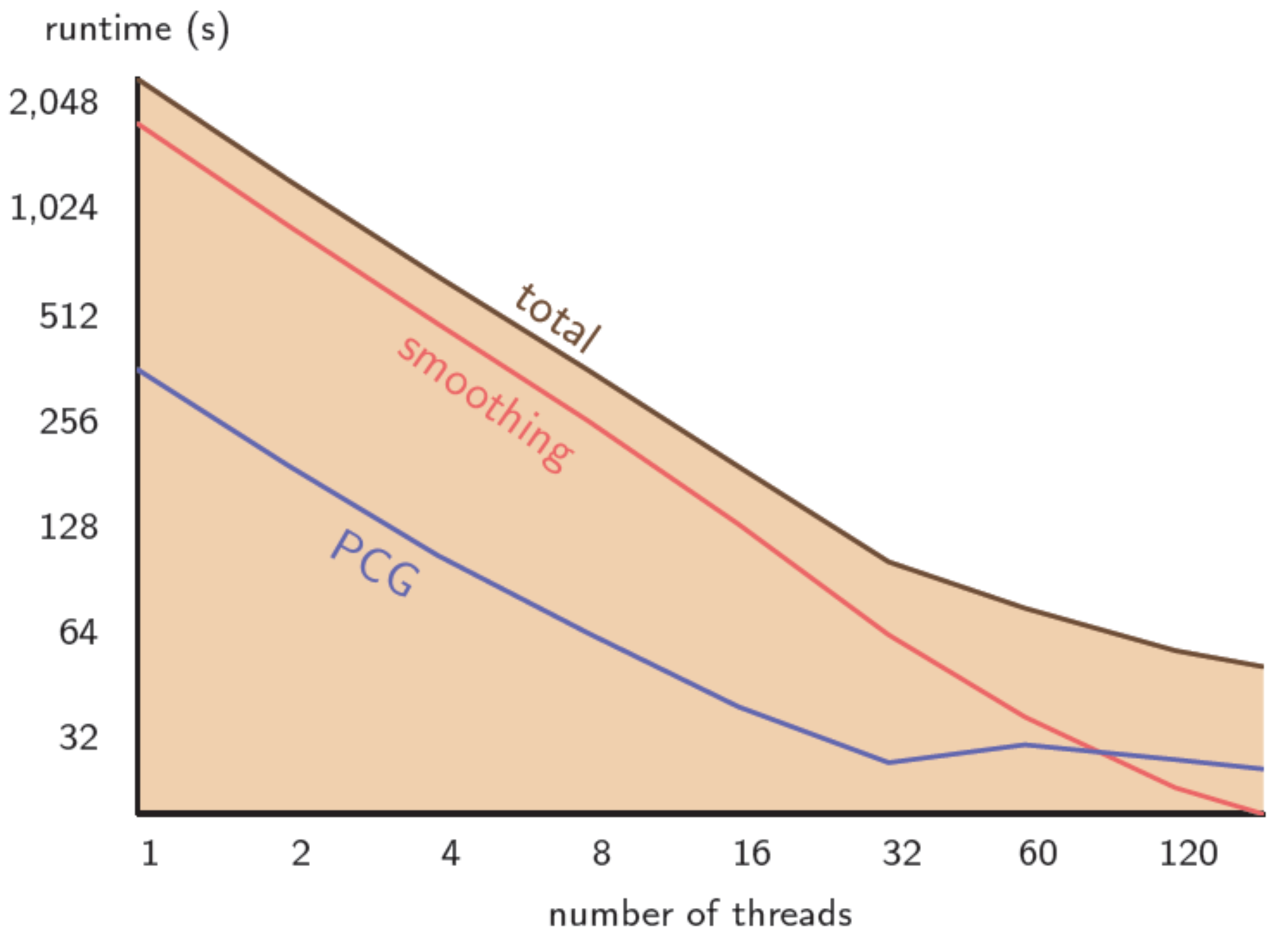
Intel Manycore Integrated Core (MIC), Knights Corner

- **60 cores per card**
- **4 hardware threads / core**
- **512-bit SIMD Vector = 8 DP FLOPS / cycle**
- **L2 private cache, coherence via bidirection**





Time breakdown of the AMG cycle



PCG thread-friendly optimization

- **Goal: minimize threads synchronization**

Algorithm 1

while not converged **do**

  $\rho \leftarrow \sigma$

  omp parallel for : $w \leftarrow Ap$

  omp parallel for : $\tau \leftarrow w \cdot p$

  $\alpha \leftarrow \rho / \tau$

  omp parallel for : $x \leftarrow x + \alpha p$

  omp parallel for : $r \leftarrow r - \alpha w$

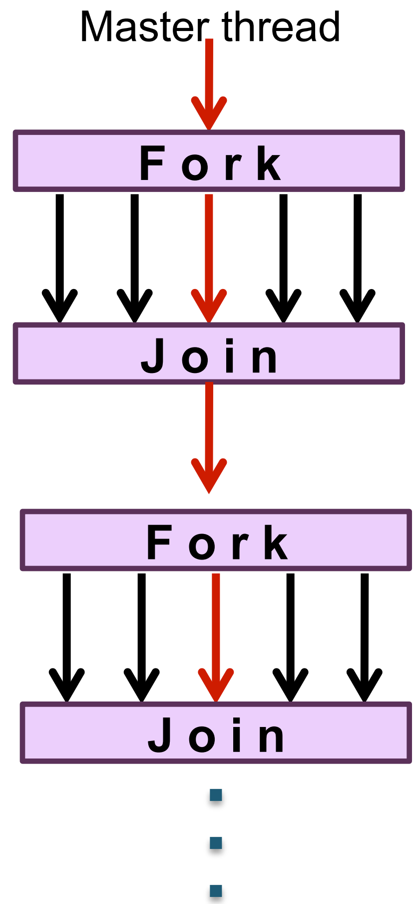
  omp parallel for : $\mathrm{z} \leftarrow \mathrm{M}^{-1} r$

  omp parallel for : $\sigma \leftarrow z \cdot r$

  $\beta \leftarrow \sigma / \rho$

  omp parallel for : $p \leftarrow z + \beta p$

end while



Improve PCG: *Algorithm 2 (omp-for-all)*

omp parallal $\mapsto$ single parallel region

  while not converged **do**

    omp single : $\tau \leftarrow 0.0$ $\mapsto$ implied barrier

    omp single nowait : $\rho \leftarrow \sigma, \sigma \leftarrow 0.0$

    omp for nowait : $w \leftarrow Ap$

    omp for reduction : $\tau \leftarrow w \cdot p$ $\mapsto$ implied barrier

    $\alpha \leftarrow \rho / \tau$

    omp for nowait : $x \leftarrow x + \alpha p$

    omp for nowait : $r \leftarrow r - \alpha w$

    omp for nowait : $z \leftarrow \mathrm{M}^{-1} r$

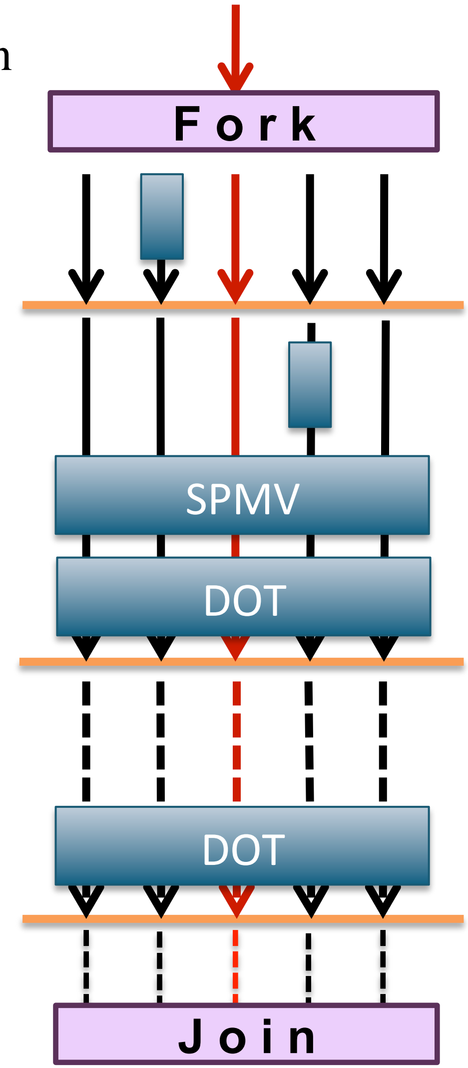
    omp for reduction : $\sigma \leftarrow z \cdot r$ $\mapsto$ implied barrier

    $\beta \leftarrow \sigma / \rho$

    omp for nowait : $p \leftarrow z + \beta p$

  end while

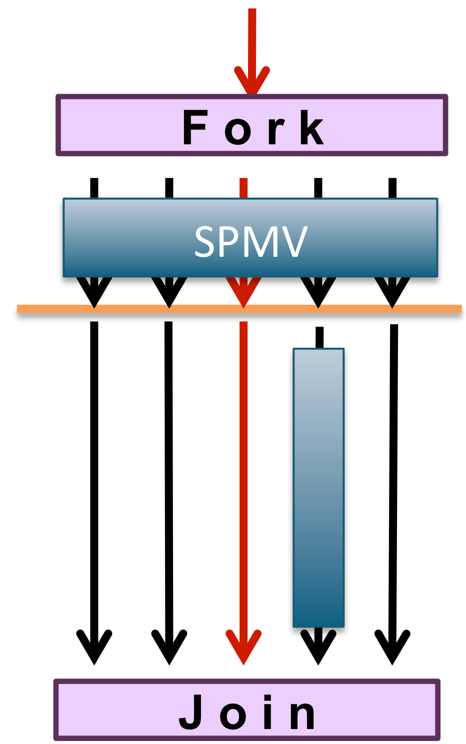
end omp parallel



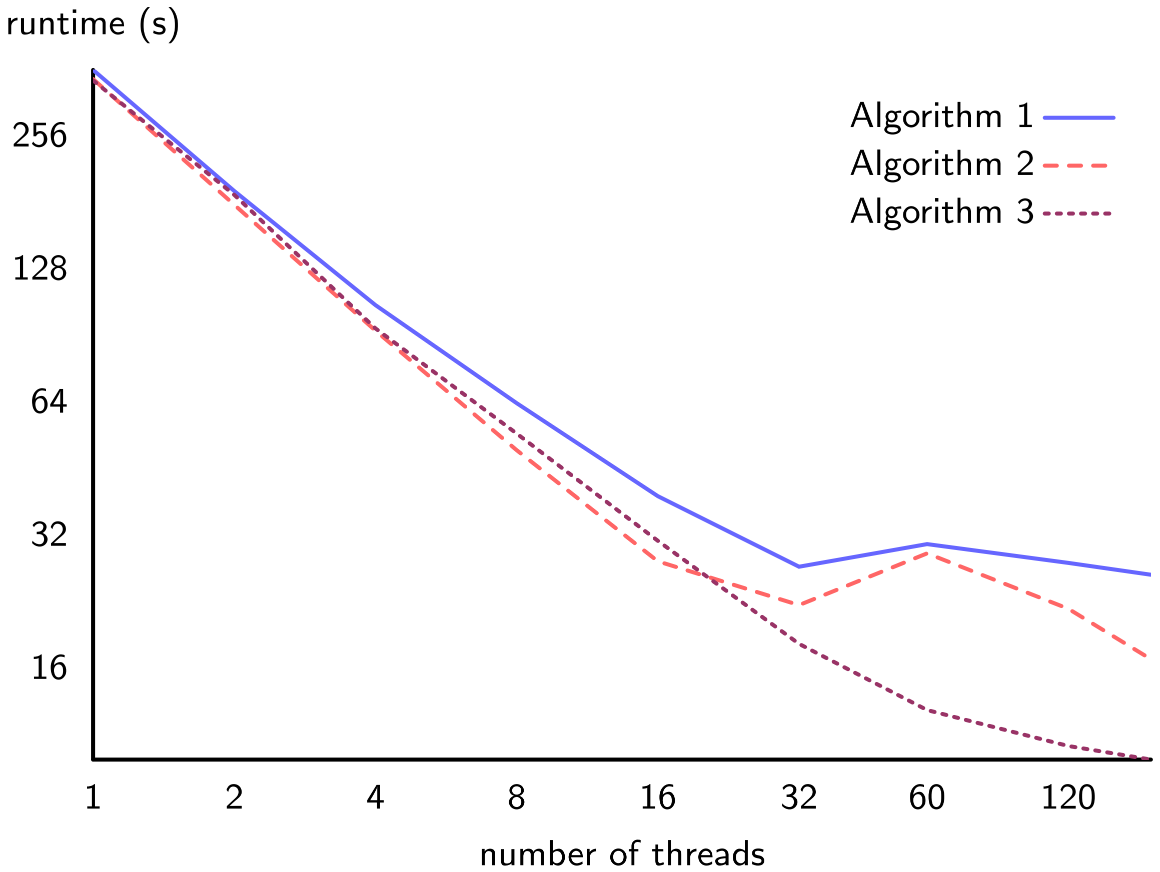
U.S. DEPARTMENT OF ENERGY | Office of Science
BERKELEY LAB

Improve PCG: *Algorithm 3 (omp-for-spmv)*

omp parallal $\mapsto$ single parallel region
  while not converged **do**
    omp for : $w \leftarrow Ap$
    omp single
      $\tau \leftarrow w \cdot p$
      $\alpha \leftarrow \rho / \tau$
      $x \leftarrow x + \alpha p$
      $r \leftarrow r - \alpha w$
      $z \leftarrow M^{-1} r$
      $\rho \leftarrow \sigma$
      $\sigma \leftarrow z \cdot r$
      $\beta \leftarrow \sigma / \rho$
      $p \leftarrow z + \beta p$
    end omp single
  end while
end omp parallel



PCG improved runtime



Roofline performance modeling (collaborating with SUPER SciDAC Institute)

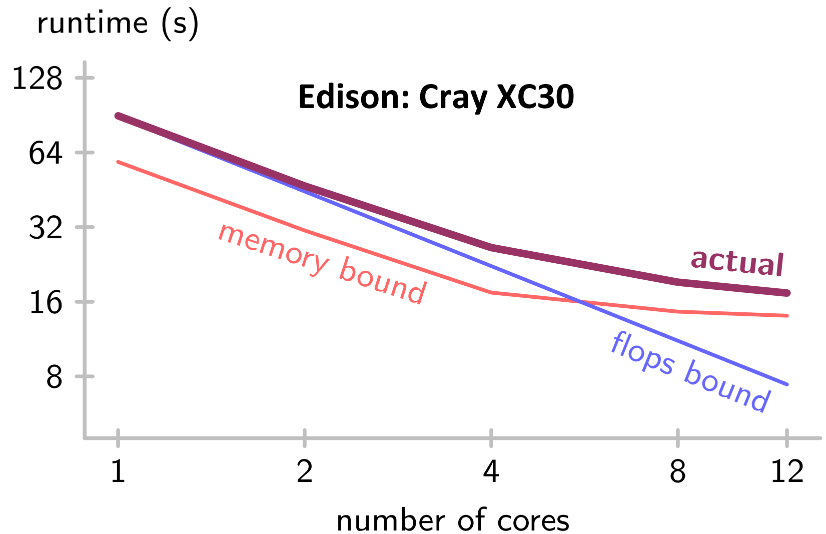
- Run time = MAX (memory access time, flops time)

stage	bytes	flops
pre- and post-smooth	$(3\nu+1)(12\,\mathbf{nza}+3\cdot 8n)$	$2(3\nu+1)(\mathbf{nza}+2n)$
restrict	$12\,\mathbf{nza}+12\,\mathbf{nzp}+3\cdot 8n$	$2(\mathbf{nza}+\mathbf{nzp})$
coarse solve (PCG/J)		
multiply by A_c	$12\,\mathbf{nzc}$	$2\,\mathbf{nzc}$
preconditioner	$2\cdot 8n_c$	n_c
vector operations	$5\cdot 8n_c$	$2\cdot 5n_c$
interpolate	$12\,\mathbf{nzp}+8n$	$2\,\mathbf{nzp}$
termination criterion	$12\,\mathbf{nza}+4\cdot 8n$	$2(\mathbf{nza}+n)$

Roofline performance modeling (collaborating with SUPER SciDAC Institute)

- Run time = MAX (flops time, memory access time)

stage	bytes	flops
pre- and post-smooth	$(3\nu+1)(12\,\mathtt{nza} + 3\cdot 8n)$	$2(3\nu+1)(\mathtt{nza}+2n)$
restrict	$12\,\mathtt{nza} + 12\,\mathtt{nzp} + 3\cdot 8n$	$2(\mathtt{nza}+\mathtt{nzp})$
coarse solve (PCG/J)		
multiply by A_c	$12\,\mathtt{nzc}$	$2\,\mathtt{nzc}$
preconditioner	$2\cdot 8n_c$	n_c
vector operations	$5\cdot 8n_c$	$2\cdot 5n_c$
interpolate	$12\,\mathtt{nzp} + 8n$	$2\,\mathtt{nzp}$
termination criterion	$12\,\mathtt{nza} + 4\cdot 8n$	$2(\mathtt{nza}+n)$



Roofline bound gap

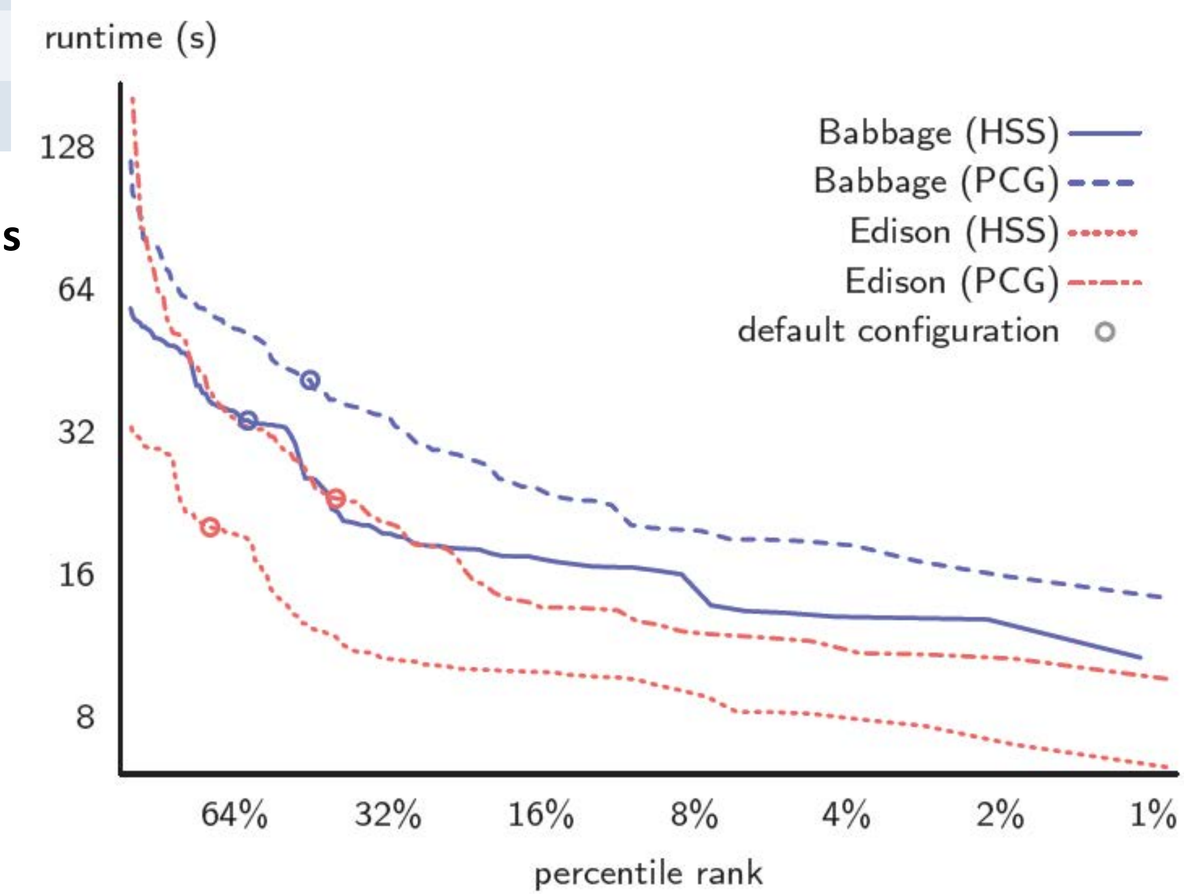
Edison: 12-cores	Coarse PCG	Coarse HSS
Gap from bound	23%	31%

A. Druinsky, P. Ghysels, X.S. Li, O. Marques, S. Williams, A. Barker, D. Kalchev, P. Vassilevski,. "Comparative Performance Analysis of Coarse Solvers for Algebraic Multigrid on Leading Multicore Architectures", PPAM 2015.

Performance variation in large paramater space

Parameters	Values
Coarse solver	PCG, HSS
Elements-per-agglogerate	64, 128, 256, 512
υ_P (poly. degree)	0, 1, 2
$\upsilon_{M^{-1}}$ (poly. degree)	
Θ (spectral tol.)	

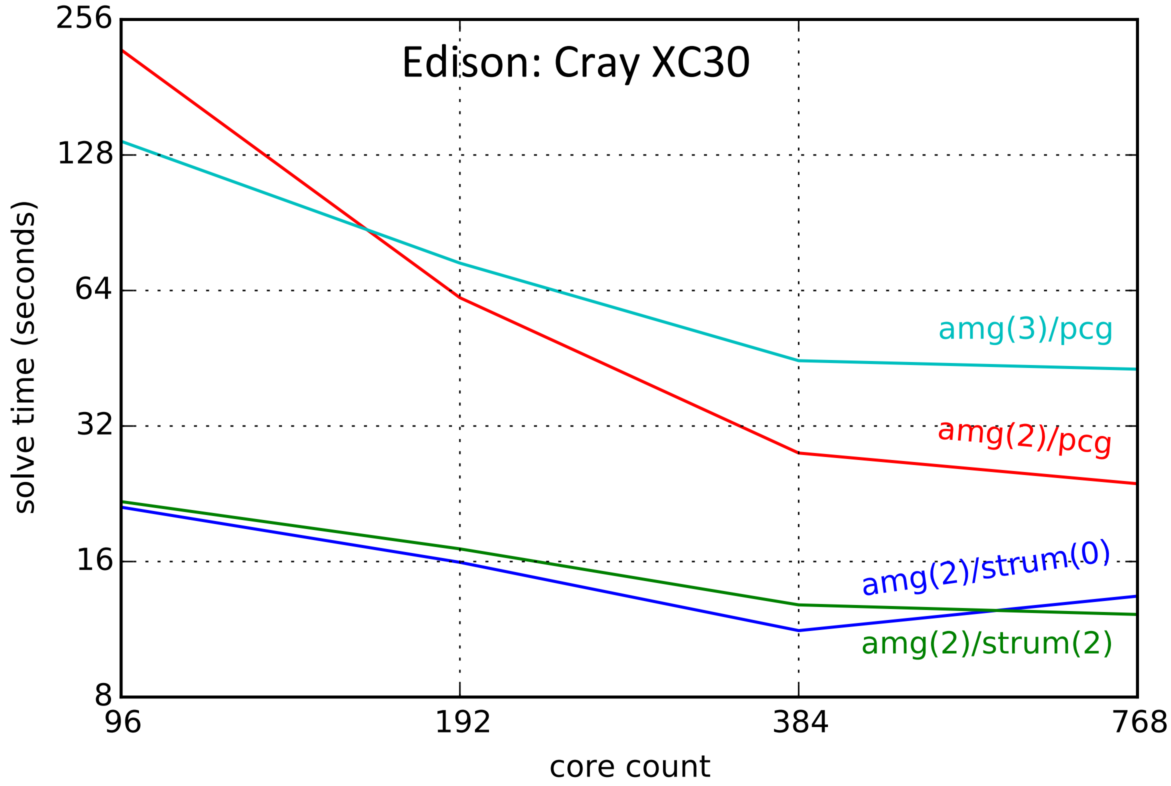
216 configurations



U.S. DEPARTMENT OF ENERGY | Office of Science

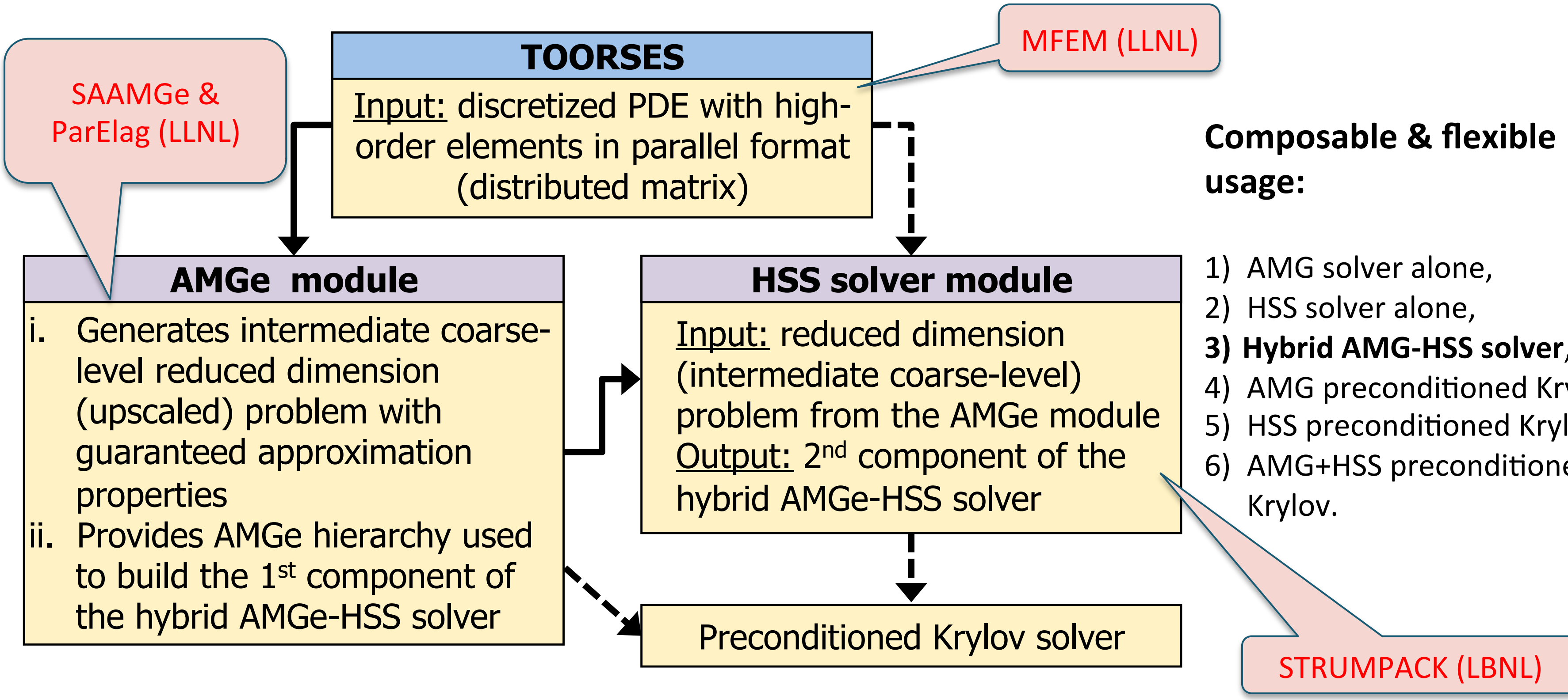
Parallel performance of multilevel AMGe + HSS

- **SPE10, 8 million DOFs**
- **Speed up MG time up to 6.4x after HSS replaced PCG as coarse level solver**



TOORSES software stack

(Towards Optimal Order Resilient Solvers at Extreme Scale)



Composable & flexible usage:

1) AMG solver alone,
2) HSS solver alone,
3) **Hybrid AMG-HSS solver**,
4) AMG preconditioned Krylov,
5) HSS preconditioned Krylov,
6) AMG+HSS preconditioned Krylov.

- MFEM: http://mfem.org
- SAAMGe, ParElag: https://myconfluence.llnl.gov/display/AMGE/AMGe
- STRUMPACK: http://portal.nersc.gov/project/sparse/strumpack

Summary, on-going work

- **Fix scalability issues in the multilevel MPI code**
  – Comprehensive performance evaluation
- **Solvers' parameters greatly influence performance**
  – Develop autotuning framework to help parameter selection
- **Initial success of two-grid, single node roofline model to understand performance limit**
  – Extend to multilevel code, multiple nodes with MPI communication
  – Improve roofline model for factorization algorithms