Scalable Linear Solvers for GPU-accelerated heterogeneous supercomputers

Pasqua D'Ambra
Institute for Applied Computing, National Research Council (IAC-CNR)
and CINI Lab on HPC-KTT

pasqua.dambra@cnr.it

2024 Frontiers of High-Performance Computing
in Modeling and Simulation
July 4-5, 2024

Greetings

HPCTeam@IAC

- Massimo Bernaschi (IAC-CNR), IT
- Mauro G. Carrozzo (IAC-CNR), IT
- Alessandro Celestini (IAC-CNR), IT
- Fabio Durastante (Univ. of Pisa and IAC-CNR), IT
- Salvatore Filippone (Univ. of Rome Tor-Vergata and IAC-CNR), IT
- Giacomo Piperno (IAC-CNR), IT

Collaborations

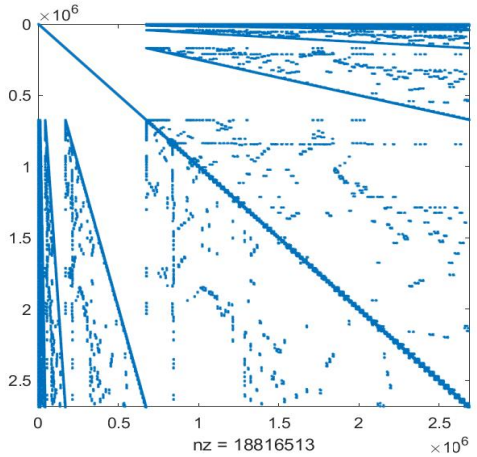
- Mahantesh M. Halappanavar and S M Ferdous , PNNL (Richland, WA), USA
- Stefano Massei, (Univ. of Pisa), IT
- Alex Pothen, Purdue University (West Lafayett, IN), USA
- Stephen Thomas, NREL, (Golden, CO), USA
- Panayot S. Vassilevski, Portland State University (Portland, OR), USA
- Ludmil Zikatanov, The Penn State University, PSU (State College), USA

What we want to solve

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathcal{R}^{n \times n} \text{ (s.p.d.)} \quad \mathbf{x}, \mathbf{b} \in \mathcal{R}^n$$

$$n \quad \textbf{large}$$

$$\text{sparsity degree} = 1 - \frac{nnz}{n^2} \approx 1$$



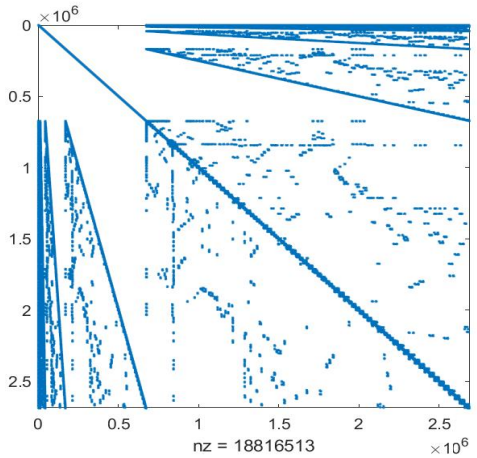
often the most time consuming computational kernel in many areas of Computational/Data Science

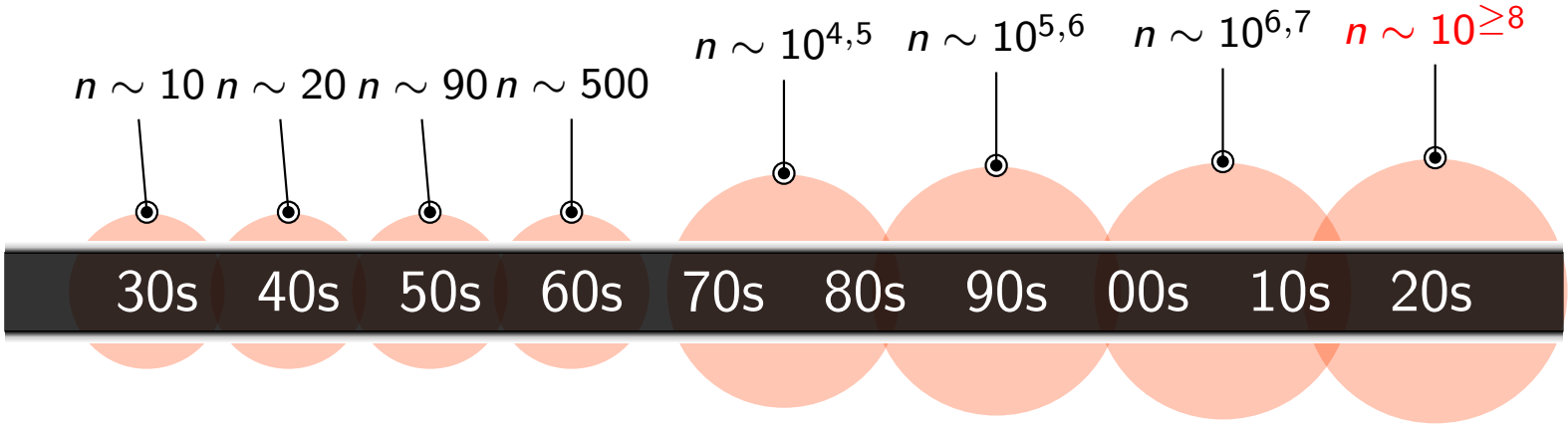
What we want to solve

$A\mathbf{x} = \mathbf{b}, \quad A \in \mathcal{R}^{n \times n}$ (s.p.d.) $\mathbf{x}, \mathbf{b} \in \mathcal{R}^n$

n **large**

$$\text{sparsity degree} = 1 - \frac{nnz}{n^2} \approx 1$$





The exascale challenge: using current supercomputers, targeting systems doing 10^{18} Flops, to solve problems with tens of trillions (10^{13}) dofs

Energy oriented Center of Excellence (EuroHPC-JU EoCoE)

Wind Models

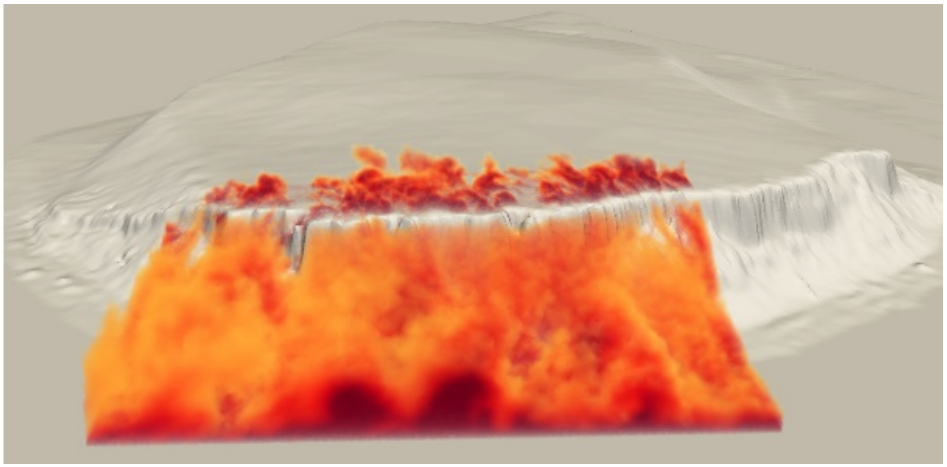
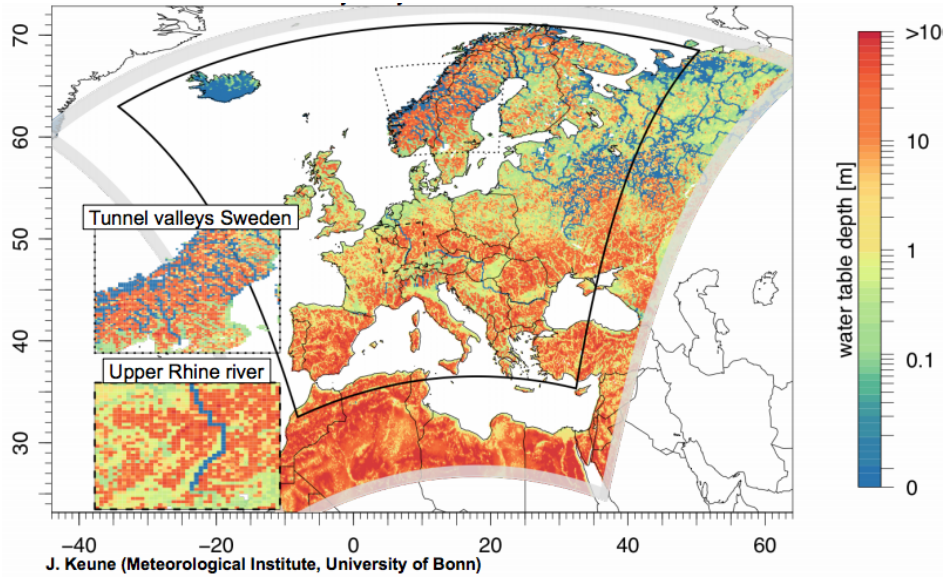


Image credits H. Owen and G. Marin, BSC

- Navier-Stokes equations,
- Euler equations,
- Turbulence models,
- ...

Regional Hydrological Models



- Darcy equation,
- Richards equation,
- Equations for overland flow
- ...

Target dofs: $n > 10^{13}$

Where we want to run[1]

	System	Cores	Rmax (PFlops)	**HPCG** (PFlops)
⋮	⋮	⋮	⋮	
7	Leonardo	1,824,768	241.20	3.1
8	MareNostrum 5 ACC	663,040	175.30	1.2
⋮	⋮	⋮	⋮	
21	Juwels Booster	449,280	44.12	1.3

Leonardo - Cineca

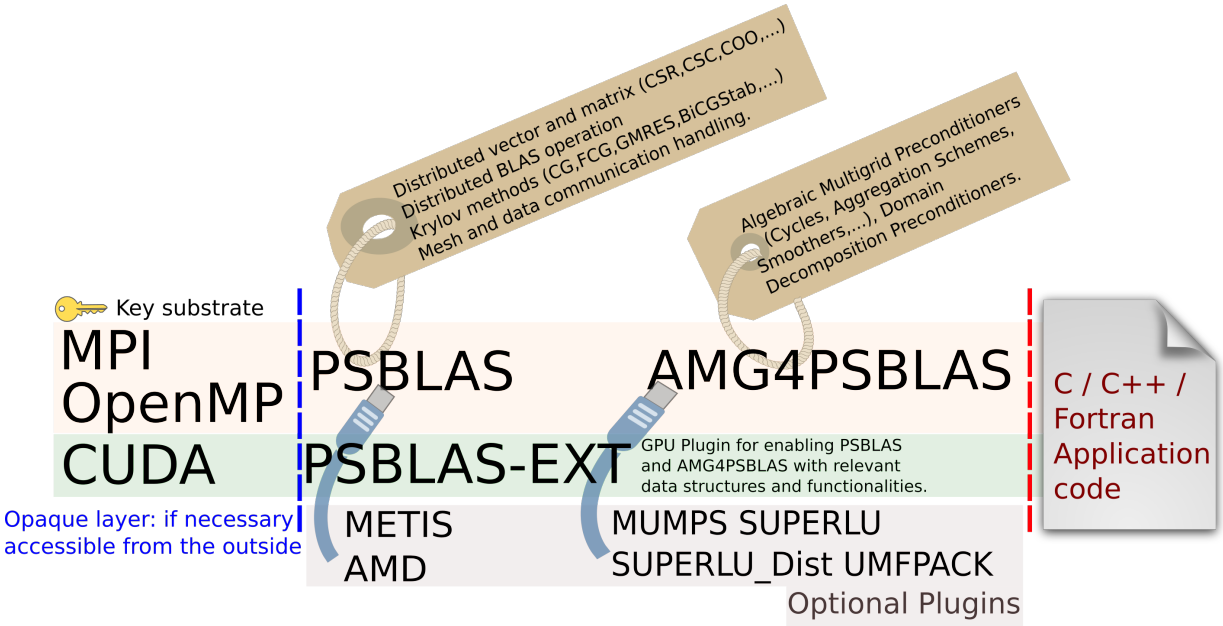
Juwels - JSC

- Computers with thousands of CPU cores and GPU accelerators
- Deep memory hierarchies and hybrid forms of parallelism/programming models&tools (MPI, OpenMP, OpenACC, Cuda, ...)

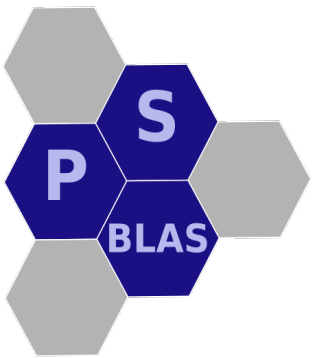
HPCG obtain a very small percentage ($\approx 1.3\%$ on Leonardo) of the peak performance!

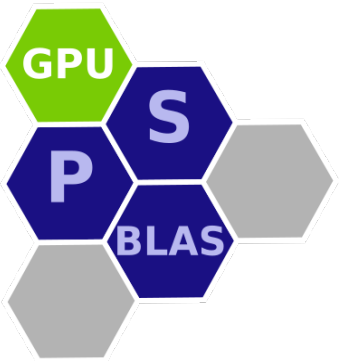
[1] TOP500 list, June 2024 – `https://www.top500.org`

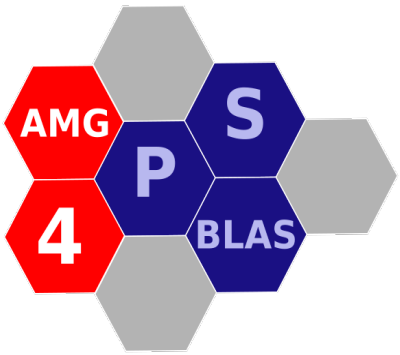
Parallel Sparse Computation Toolkit



recognized as "Excellent Science Innovation"
by the EU Innovation Radar







(Algebraic) MultiGrid methods

V-cycle$(l, nlev, A^l, b^l, x^l)$

if $(l \neq nlev)$ then

$x^l = x^l + (M^l)^{-1}(b^l - A^l x^l)$

$b^{l+1} = (P^l)^T (b^l - A^l x^l)$

$x^{l+1} = \text{V-cycle}(l+1, A^{l+1}, b^{l+1}, 0)$

$x^l = x^l + P^l x^{l+1}$

$x^l = x^l + (M^l)^{-T}(b^l - A^l x^l)$

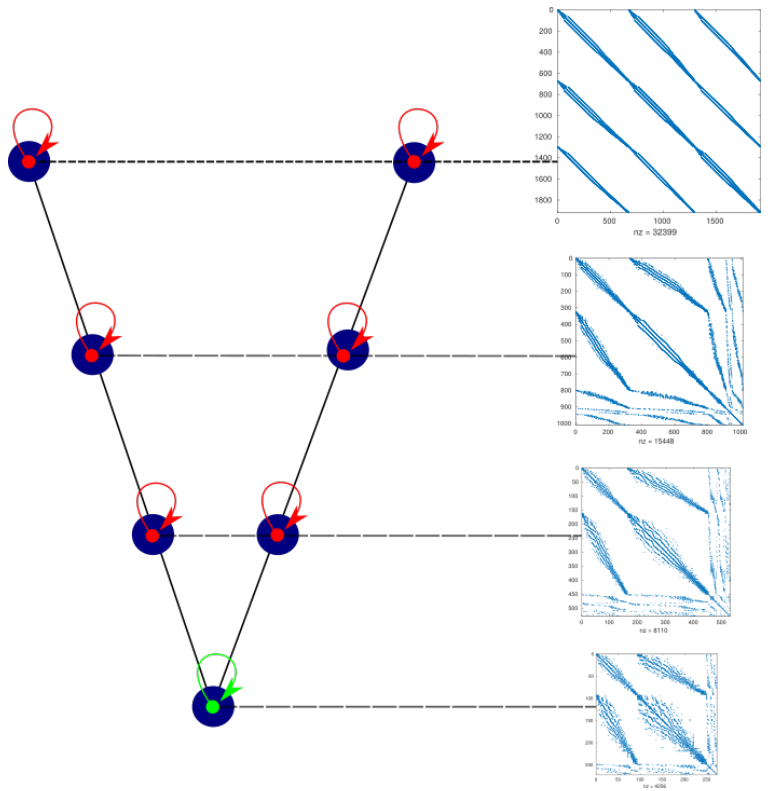
else

$x^l = (A^l)^{-1} b^l$

endif

return x^l

end



Smoother

$$M^l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$$

"damping high frequencies"

Prolongator

$$P^l : \mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_l}$$

"transferring low frequencies"

MultiGrid Convergence

Theorem (McCormick 1985, Vassilevski 2008)

If M^l is a contraction at each level l, i.e., $\|I - (M^l)^{-1}A^l\|_{A^l} < 1$, the V-cycle preconditioner B defined as the multiplicative composition of the iteration matrix

$$I - (B^l)^{-1}A^l = (I - (M^l)^{-T}A^l)(I - P^l((P^l)^T A^l P^l)^{-1}(P^l)^T A^l)(I - (M^l)^{-1}A^l)$$

has the following error bound:

$$\|E\|_A^2 = \|I - B^{-1}A\|_A^2 \leq 1 - \frac{1}{C} \quad \textit{with}$$

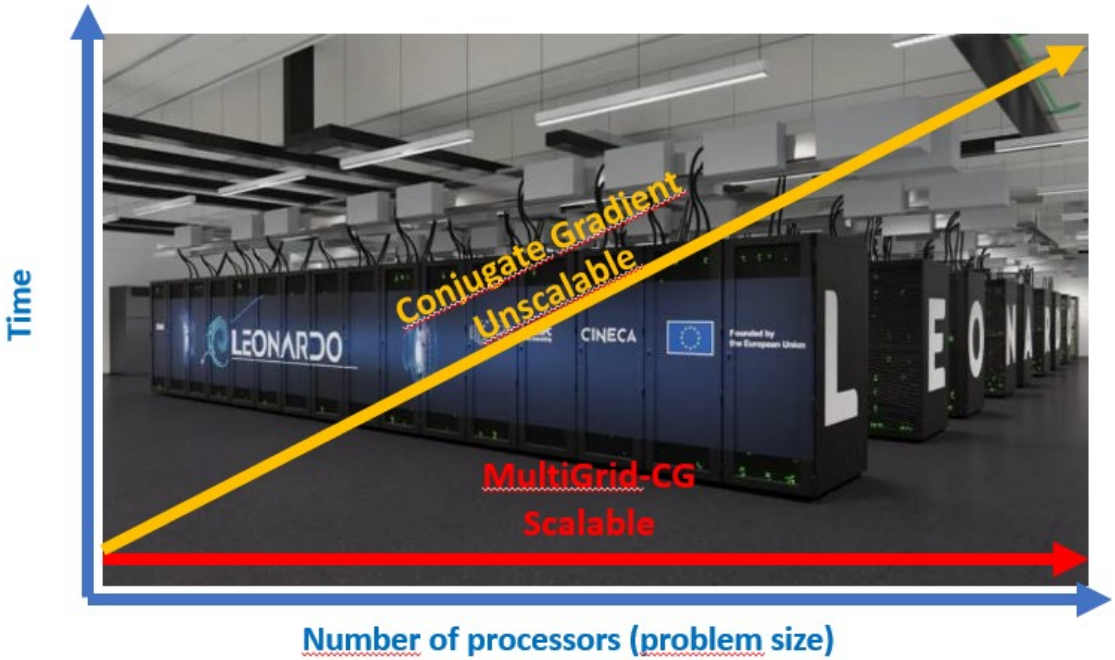
$$C = max_l C^l$$

where $C^l = sup_{v \in \mathcal{R}ange(P^l)^{\perp_A} \setminus 0} \frac{\|v\|_{\tilde{M}^l}^2}{\|v\|_A^2} \geq 1$ *is the strong approximation constant and* $\tilde{M}^l = M^l(M^l + (M^l)^T - A^l)^{-1}(M^l)^T$ *is the symmetrized smoother.*

Optimal Convergence (independent of problem size and number of levels)

Scalable (AMG) preconditioners

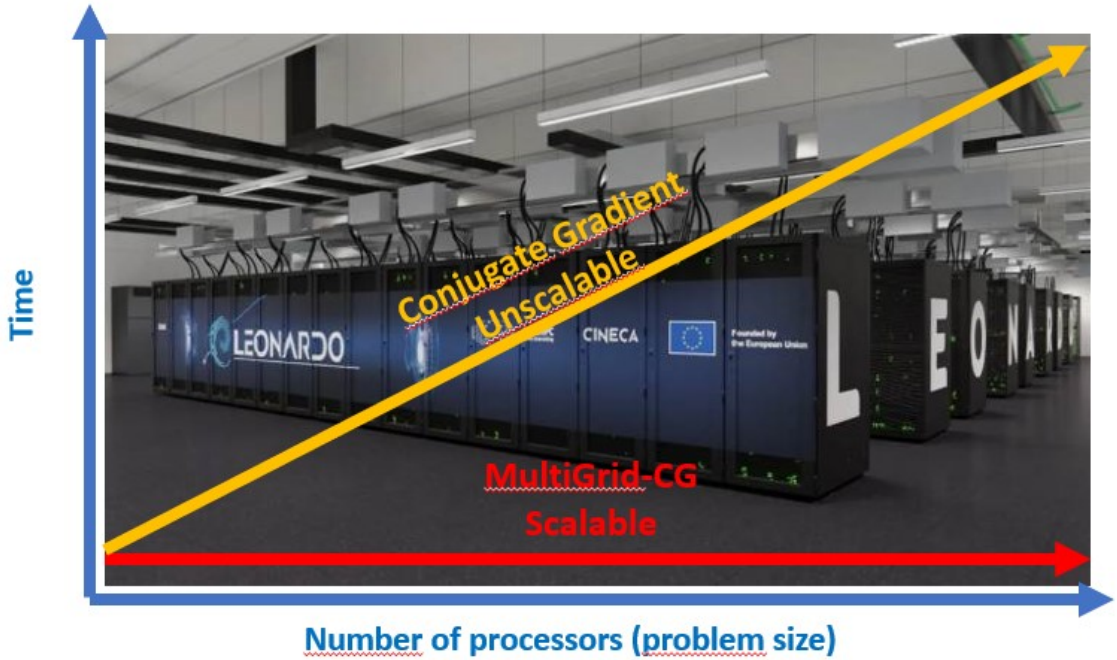
(inspired by Rob Falgout)



- Optimal complexity: AMG can be optimal ($\mathcal{O}(nnz)$ flops), then have good scalability potential

Scalable (AMG) preconditioners

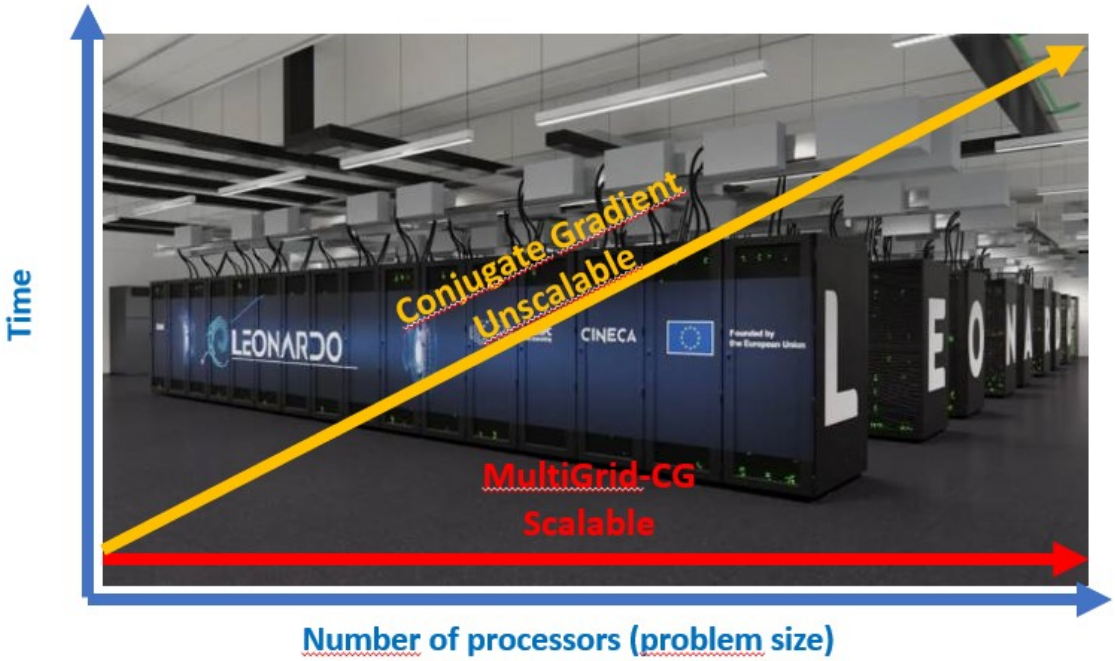
(inspired by Rob Falgout)



- Optimal complexity: AMG can be optimal ($\mathcal{O}(nnz)$ flops), then have good scalability potential
- Optimal convergence: $\|E\|_A^2 < 1$ being independent of nnz (Algorithmic scalability)

Scalable (AMG) preconditioners

(inspired by Rob Falgout)



Optimal complexity and optimal convergence are not sufficient for HPC!

Implementation scalability: B should be composed of local actions essentially based on a "hierarchy" of sparse matrix-vector products

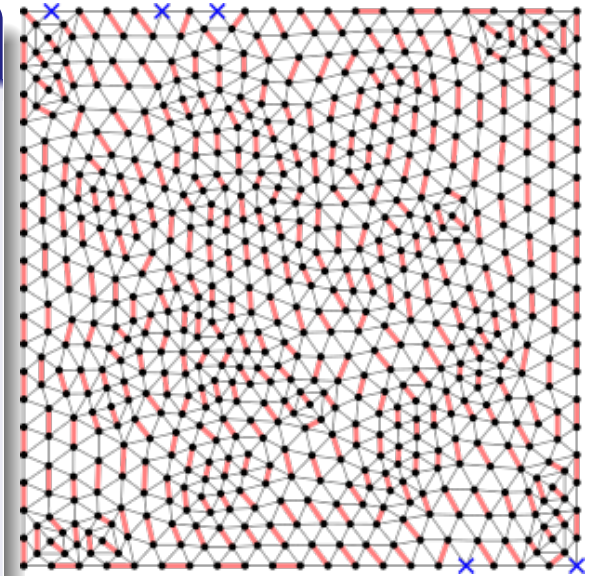
Parallel AMG setup

CMATCH algorithm

Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight (smooth) vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{A}$:

$$(\hat{A})_{i,j} = \hat{a}_{i,j} = 1 - \frac{2a_{i,j}w_i w_j}{a_{i,i}w_i^2 + a_{j,j}w_j^2},$$

- a *matching* $\mathcal{M}$ is a set of pairwise non-adjacent edges, containing no loops;
- a maximum product matching maximizes the product of the weights of its edges $e_{i \mapsto j}$.



Divide the index set into matched vertices $\mathcal{I} = \bigcup_{i=1}^{n_p} \mathcal{G}_i$, with $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ if $i \neq j$,

- Increase coarsening ratio by performing more than one sweep of matching per level
- Increase regularity of P_{l+1}^{l} with smoothed prolongator by applying one step of Jacobi method

PD et al., AMG Preconditioners for Linear Solvers towards Extreme Scale, SIAM Journal on Scientific Computing, 43, 5, 2021

Parallel Smoothers

Let M be the spd (convergent) ℓ_1-Jacobi smoother:

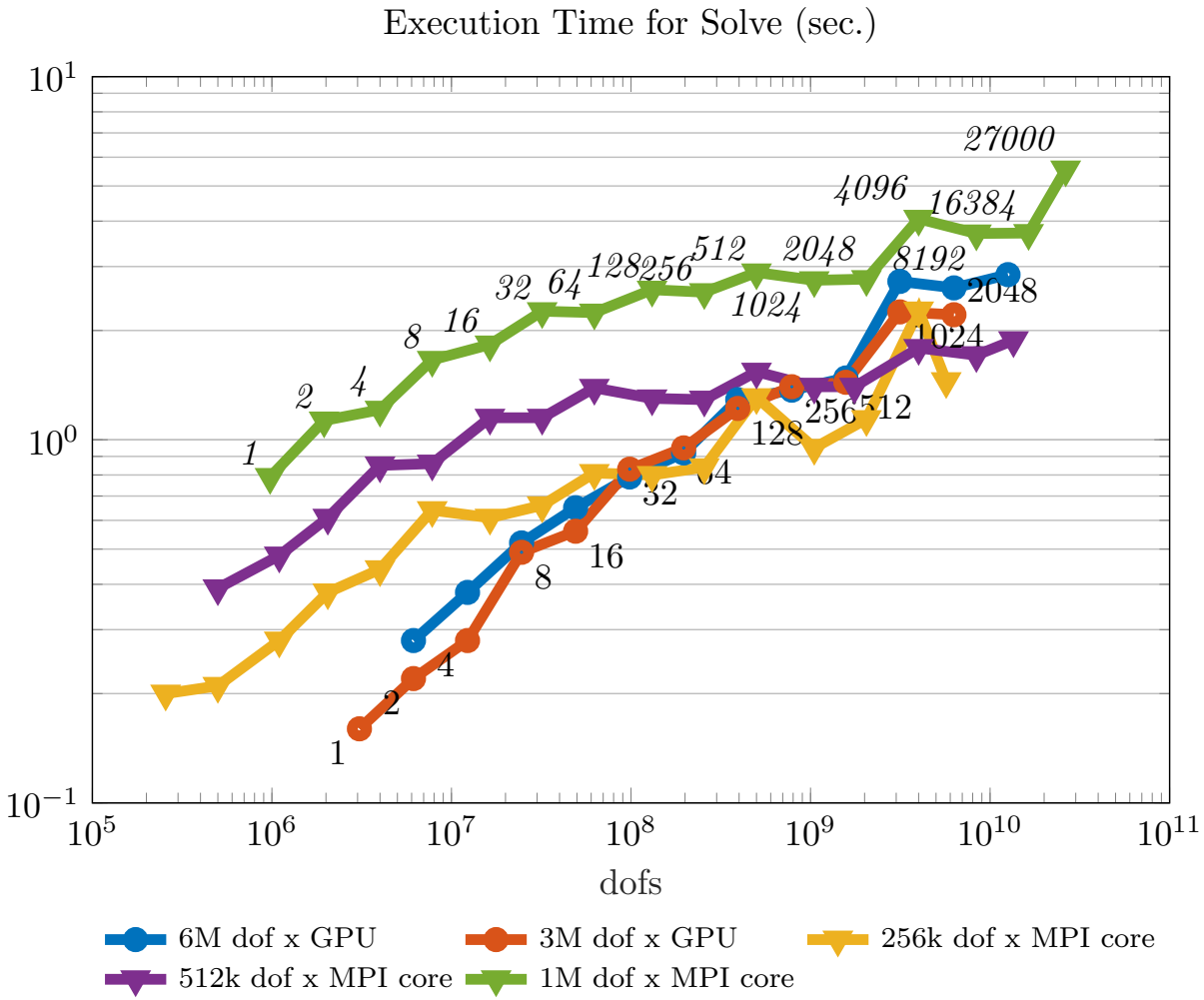
$$G = (I - M^{-1}A), \qquad \begin{array}{l} M = \mathrm{diag}(M_{ii})_{i=1,\ldots,n} \\ M_{ii} = a_{ii} + \sum_{j \neq i} |a_{ij}| \end{array}$$

Pros: simple and cheap to setup, only based on sparse matrix-vector product and local vector updates well suited for high-throughput SIMD processors

Cons: larger approximation constant than block-Jacobi version of Gauss-Seidel (Hybrid GS); in our AMG setting the constant is ≈ 4 time larger for systems coming from 3D Poisson problem

Parallel Smoothers

Some results on Piz Daint (ranked 43th): MPI-HGS vs MPI/CUDA-I1Jac



using GPUs saves up to $\approx 50\%$

in solve time and energy consumption for 10 billion dofs

Parallel Smoothers

Polynomial accelerators (Adams et al. 2003, Kraus et al. 2012)

$$G = p_k((M^l)^{-1}A^l), \text{ for } p_k(x) \in \Pi_k[x]$$

$$\text{s.t. } p_k(0) = 1 \text{ and } |p_k(x)| < 1 \text{ for } 0 < x \leq 1$$

Key issue: choose polynomials to optimize V-cycle approximation constant

Parallel Smoothers

Polynomial accelerators (Adams et al. 2003, Kraus et al. 2012)

$$G = p_k((M^l)^{-1}A^l), \text{ for } p_k(x) \in \Pi_k[x]$$
$$\text{s.t. } p_k(0) = 1 \text{ and } |p_k(x)| < 1 \text{ for } 0 < x \leq 1$$

Key issue: choose polynomials to optimize V-cycle approximation constant

Theorem (Lottes, 2023)

The V-cycle error propagation matrix has following bound:

$$\|E\|_A^2 \leq max_l \frac{C^l}{C^l + (\gamma_k^l)^{-1}},$$

where C^l is the strong approximation property constant at the level l and

$$\gamma_k^l = \sup_{0<\lambda\leq 1} \frac{\lambda p_k(\lambda)^2}{1 - p_k(\lambda)^2}$$

Optimal V-cycle bound & Chebyshev polynomials

Minimax problem

$$\gamma_k := \min_{p_k(x)\in\Pi_k} \max_{x\in(0,1]} \left| \frac{x p_k(x)^2}{1 - p_k(x)^2} \right|$$

$$\text{s.t. } p_k(0) = 1 \text{ and } |p_k(x)| < 1 \text{ for } 0 < x \leq 1$$

Optimal V-cycle bound & Chebyshev polynomials

Minimax problem

$$\gamma_k := \min_{p_k(x)\in\Pi_k} \max_{x\in(0,1]} \left| \frac{x p_k(x)^2}{1 - p_k(x)^2} \right|$$

$$\text{s.t. } p_k(0) = 1 \text{ and } |p_k(x)| < 1 \text{ for } 0 < x \leq 1$$

Rewriting the minimax problem

$$\gamma_k = \min_{p_k(x)\in\Pi_k} \max_{x\in(0,1]} x \left| 1 - \frac{1}{1 - p_k(x)^2} \right|,$$

$$\text{s.t. } p_k(0) = 1 \text{ and } |p_k(x)| < 1 \text{ for } 0 < x \leq 1$$

Optimal V-cycle bound & Chebyshev polynomials

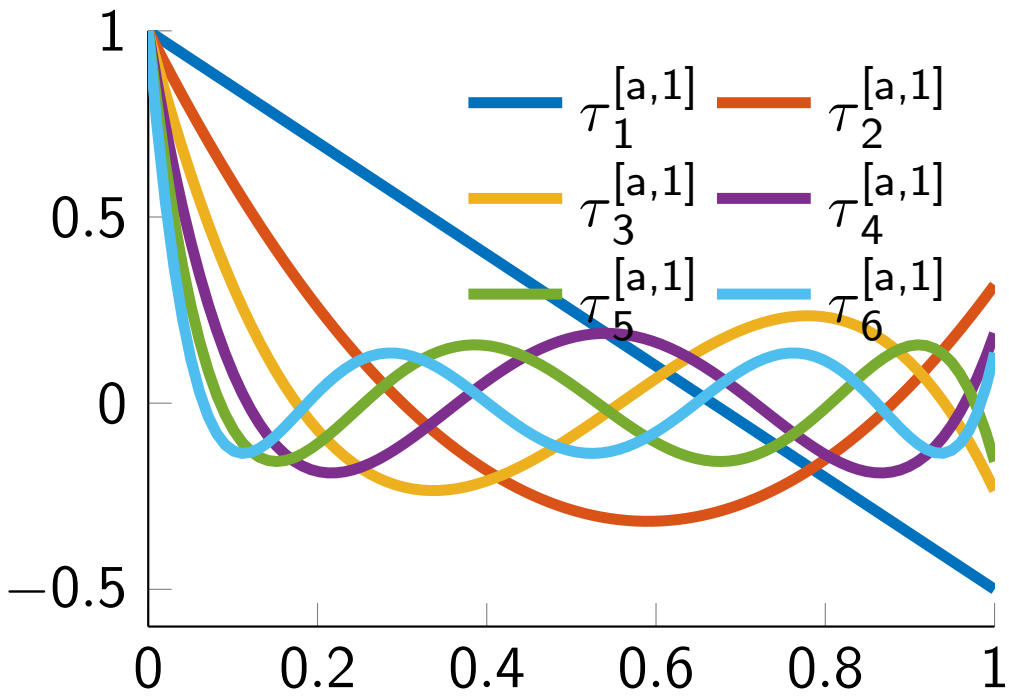
Quasi-Optimal 1st-kind Chebyshev polynomials

$$\tau_k(x) = \frac{1}{2}\left[(x+\sqrt{x^2-1})^k + (x-\sqrt{x^2-1})^k\right]$$

- $\tau_k(x)$ provides the optimal solution in the interval $[a_k, 1]$, for $a_k \in (0,1)$
- optimal values of a_k and corresponding γ_k can be numerically obtained by solving a scalar non-linear equation
- lower and upper bounds can be found for optimal a_k and γ_k
- can be applied as a simple 3-terms recurrence requiring sparse matrix-vector products and vector updates

PD et al., *Optimal Polynomial Smoothers for Parallel AMG*, SIAM ALA, Paris, May 2024.

Optimal V-cycle bound & Chebyshev polynomials

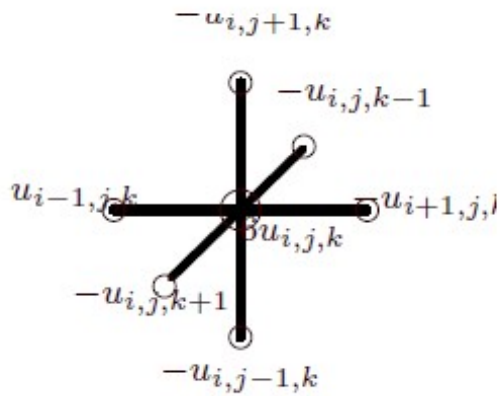


$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \mathbf{z}^{(k-1)}$$

$$\begin{cases} \mathbf{z}^{(0)} = \frac{2}{1+a^*} M^{-1}(\mathbf{b} - A\mathbf{x}^{(0)}) \ \rho_0 = \frac{1-a^*}{1+a^*} \\ \rho_k = \left(\frac{2(1+a^*)}{1-a^*} - \rho_{k-1} \right)^{-1} \\ \mathbf{z}^{(k)} = \rho_k \rho_{k-1} \mathbf{z}^{(k-1)} + \frac{4\rho_k}{(1-a^*)} M^{-1}(\mathbf{b} - A\mathbf{x}^{(k)}) \end{cases}$$

Test case: Poisson equation

$-\nabla^2 u = 1$ on unit cube, with DBC



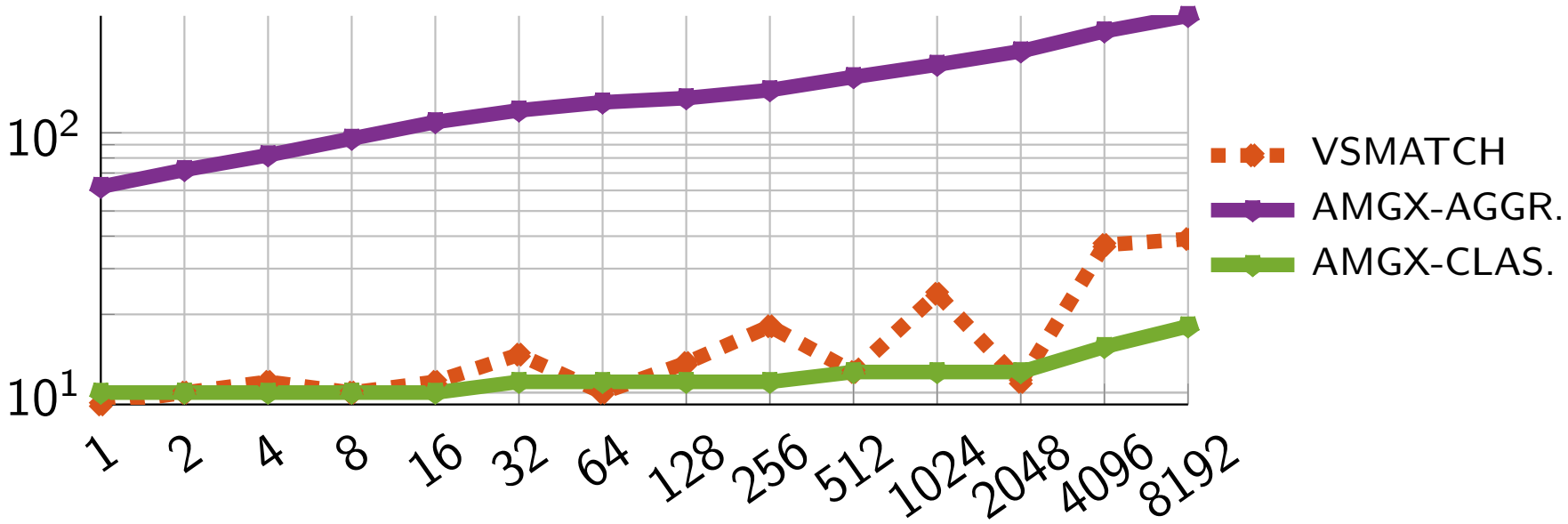
Solver/preconditioner settings

- AMG as preconditioner of CG, stopped when $\|\mathbf{r}^k\|_2/\|\mathbf{b}\|_2 \leq 10^{-6}$

  VSMATCH V-cycle for CMATCH-based AMG hierarchy with aggregates of max size 8, smoothed prolongators
- coarsest matrix size $n_c \leq 200np$, with np number of tasks (GPUs)
- 4 ℓ_1-Jacobi iterations as pre/post smoother; 40 iterations of PCG + ℓ_1-Jacobi at the coarsest level.

Platform: Leonardo booster, BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband

Iterations for Solve up to 65 billion dofs

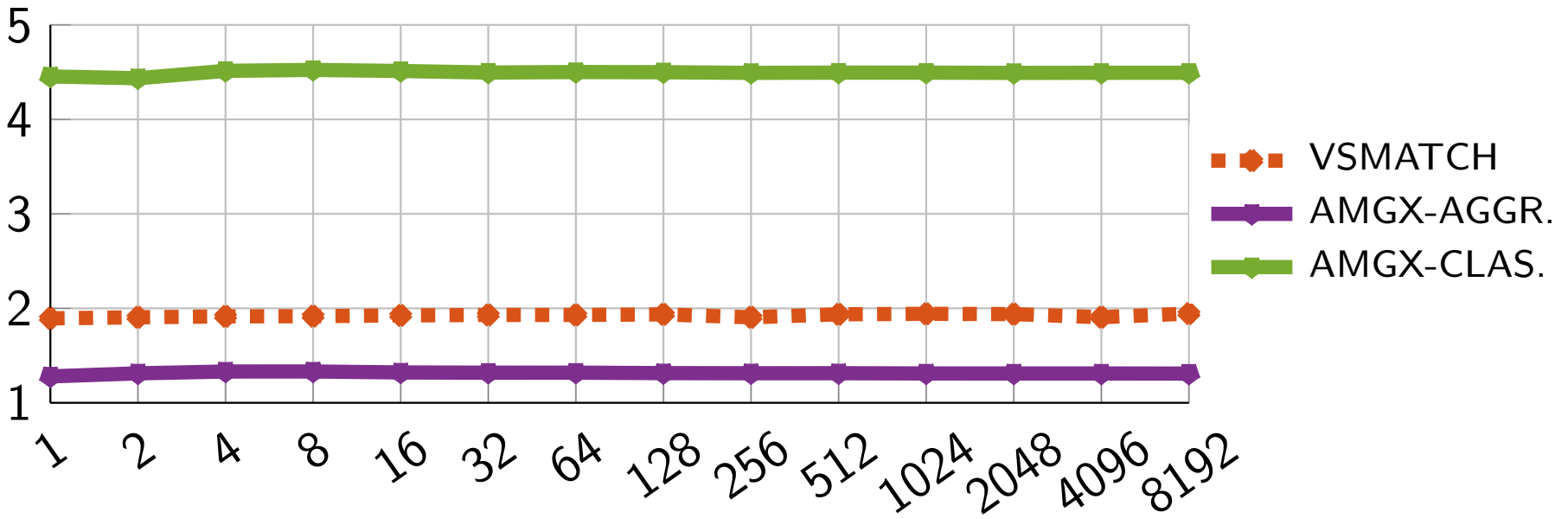


Comparison with Nvidia AmgX-PCG coupled with:

AMGX-CLAS. Ruge-Stüben coarsening, distance-2 interpolation, V-cycle, smoother ℓ_1-Jacobi (4 sweeps), ℓ_1-Jacobi (40 sweeps) coarsest solver

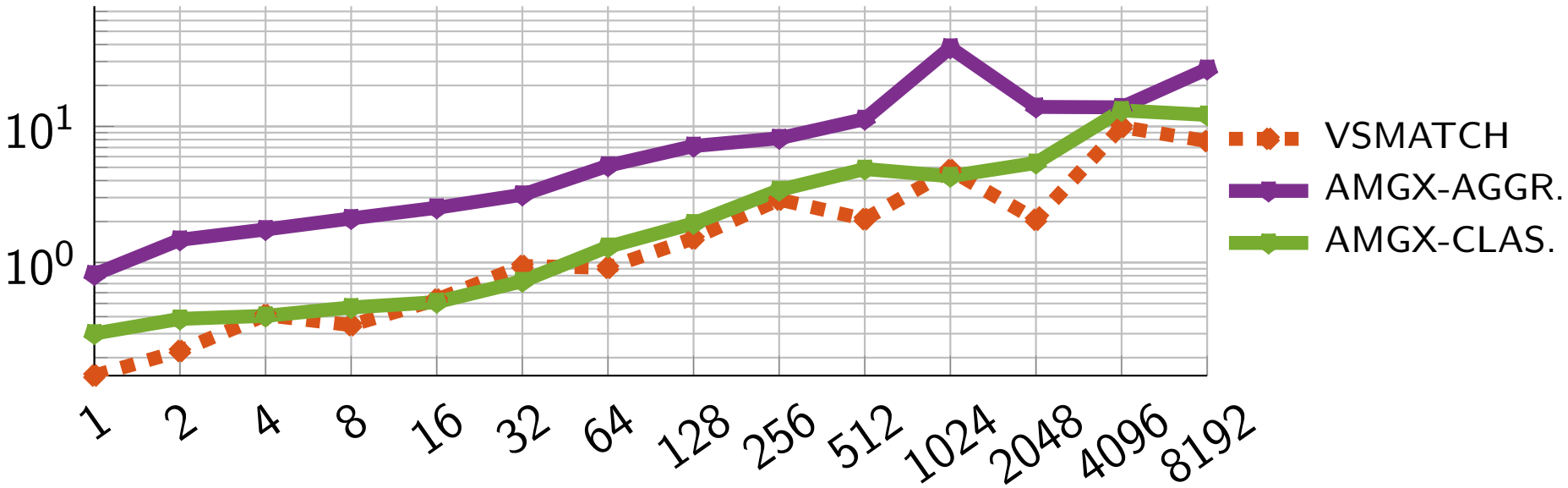
AMGX AGGR. aggregation by iterative parallel graph matching, aggregates of maximum size equal to 8, further algorithmic choices as in AMGX-CLASS

Operator Complexity

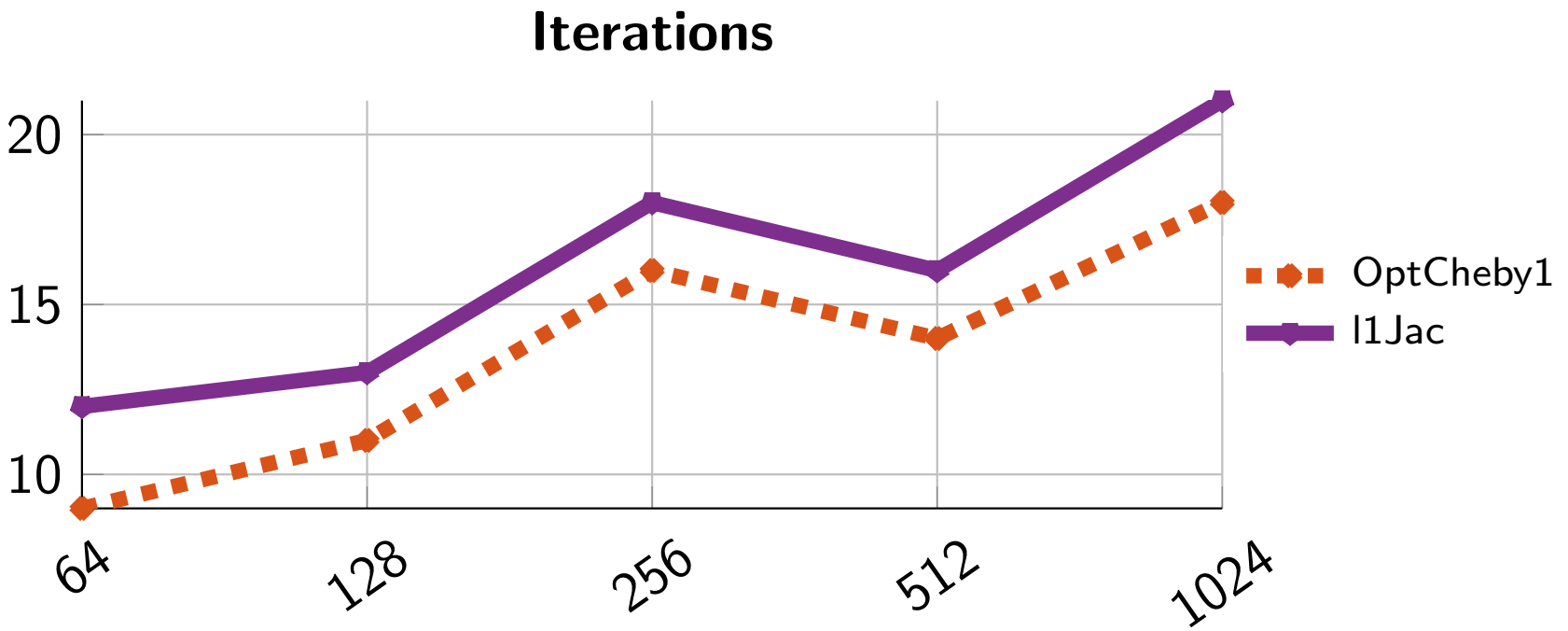


$$\frac{\sum_{l=0}^{\ell} \mathrm{nnz}(A_l)}{\mathrm{nnz}(A_0)}$$

Solve Time (sec.)

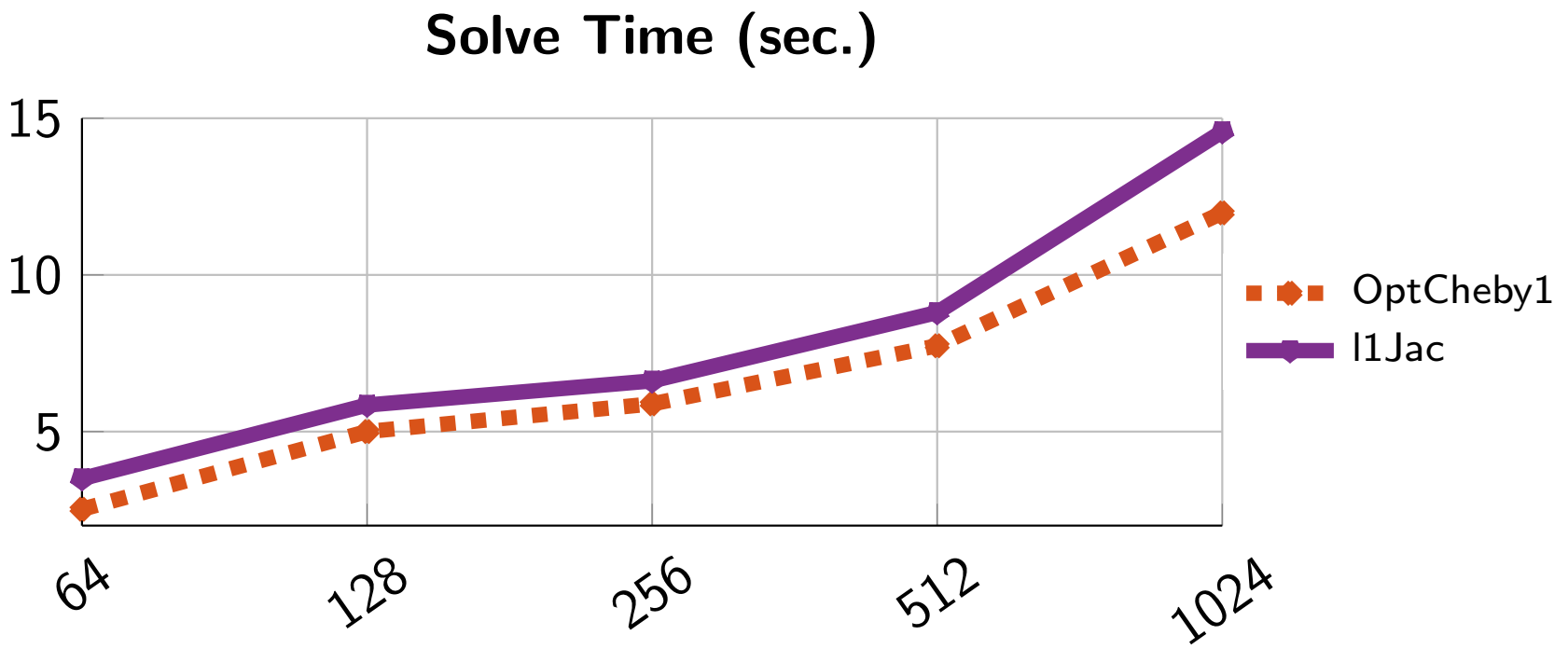


VSMATCH: Opt-1st-kind Cheb.(deg 4) vs l1-Jac up (4 it)



17% iterations savings on 1024 GPUs for 6 billion dofs

VSMATCH: Opt-1st-kind Cheb.(deg 4) vs l1-Jac up (4 it)



22% time savings on 1024 GPUs for 6 billion dofs

Some concluding remarks

- PSCToolkit demonstrates benefits in solving benchmark systems up to 65 billion dofs on up to 8192 GPUs of the Leonardo supercomputer
- Applications to CFD for sustainable energy are in:
  - PD et al., *Alya towards Exascale: Algorithmic Scalability using PSCToolkit*, Journal of Supercomputing, Vol. 80, 2024.
  - PD et al., *Why diffusion-based preconditioning of Richards equation works: spectral analysis and computational experiments at very large scale*, Numerical Linear Algebra with Applications, Vol. 31, 1, 2024.
- Improvements in methods and software design are work in progress within different projects

Thanks for Your Attention

This work was performed with support of the European Union under HORIZON-EUROHPC-JU-2023-COE-01 agreement N. 101144014 — EoCoE-III and of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR - Next Generation EU (NGEU)

Coarsening by Compatible Weighted Matching

Let $\mathbf{w} \in \mathcal{R}^n$ smooth vector, let $P_c \in \mathcal{R}^{n \times n_c}$ and $P_f \in \mathcal{R}^{n \times n_f}$ be a prolongator and a complementary prolongator, such that:

$$\mathcal{R}^n = \mathcal{R}ange(P_c) \oplus^{\perp} \mathcal{R}ange(P_f), \quad n = n_c + n_f$$

$\mathbf{w} \in \mathcal{R}ange(P_c)$: coarse space $\quad \mathcal{R}ange(P_f)$: complementary space

$$[P_c, P_f]^T A [P_c, P_f] = \begin{pmatrix} P_c^T A P_c & P_c^T A P_f \\ P_f^T A P_c & P_f^T A P_f \end{pmatrix} = \begin{pmatrix} A_c & A_{cf} \\ A_{fc} & A_f \end{pmatrix}$$

A_c : coarse matrix $\quad A_f$: hierarchical complement

Compatible Relaxation (Falgout and Vassilevski, 2004)

Good estimate of the strong approximation constant for P_c:
$\rho_f = \|I - M_f^{-1} A_f\|_{A_f}$ with $M_f = P_f^T M P_f$

Coarsening by Compatible Weighted Matching

Let $\mathbf{w} \in \mathcal{R}^n$ smooth vector, let $P_c \in \mathcal{R}^{n \times n_c}$ and $P_f \in \mathcal{R}^{n \times n_f}$ be a prolongator and a complementary prolongator, such that:

$$\mathcal{R}^n = \mathcal{R}ange(P_c) \oplus^{\perp} \mathcal{R}ange(P_f), \quad n = n_c + n_f$$

$\mathbf{w} \in \mathcal{R}ange(P_c)$: coarse space $\quad \mathcal{R}ange(P_f)$: complementary space

$$[P_c, P_f]^T A [P_c, P_f] = \begin{pmatrix} P_c^T A P_c & P_c^T A P_f \\ P_f^T A P_c & P_f^T A P_f \end{pmatrix} = \begin{pmatrix} A_c & A_{cf} \\ A_{fc} & A_f \end{pmatrix}$$

A_c : coarse matrix $\quad A_f$: hierarchical complement

Compatible Relaxation (Falgout and Vassilevski, 2004)

Good estimate of the strong approximation constant for P_c:
$\rho_f = \|I - M_f^{-1} A_f\|_{A_f}$ with $M_f = P_f^T M P_f$

Our recipe

build P_c (and P_f) by dofs **aggregation based on matching in the weighted (adjacency) graph of** A, to make A_f **as diagonally-dominant as possible**

Results: Time per Iteration