Node-level efficiency and scalability issues in iterative sparse linear solvers at scale

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

pasqua.dambra@cnr.it

31st Euromicro International Conference on PDP

March 1-3, 2023

The HPC Team at 

Projects Participants :

- Massimo Bernaschi (IAC-CNR), IT
- Mauro 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
- Lorenzo Pichetti (Univ. of Trento and IAC-CNR), IT
- Flavio Vella (Univ. of Trento and IAC-CNR), IT

Collaborations :

Mahantesh M. Halappanavar and S M Ferdous , PNNL (Richland, WA), USA (see our joint paper in PDP 2023 Proceedings book)

Alex Pothen, Purdue University (West Lafayett, IN), USA

Panayot S. Vassilevski, Portland State University (Portland, OR), USA

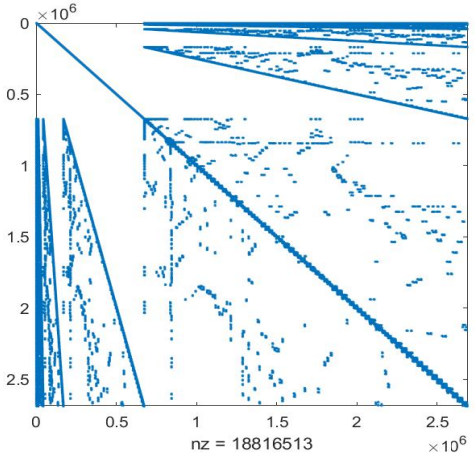
Ludmil Zikatanov, The Penn State University, PSU (State College), USA

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 **large**

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



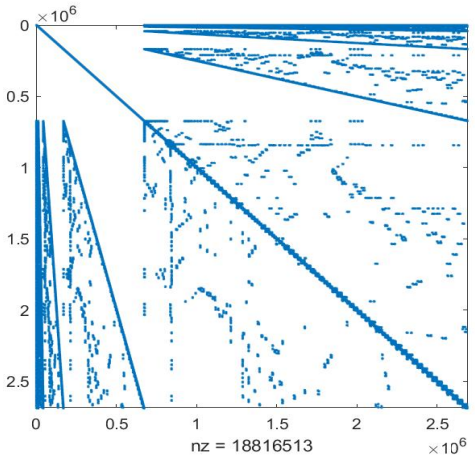
often the most time consuming computational kernel in many areas of Computational/Data Science

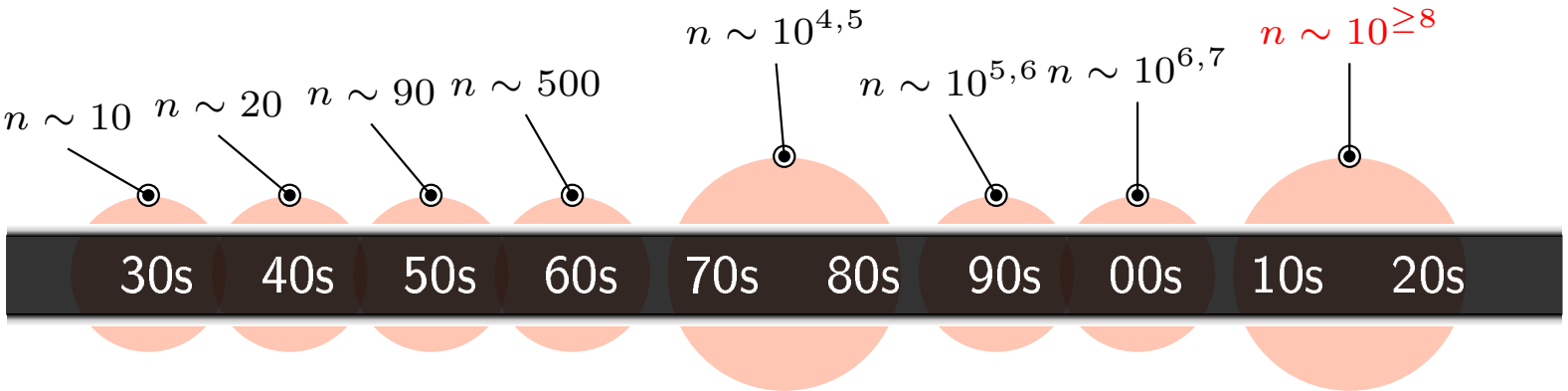
What we want to solve

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

n **large**

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





The exascale challenge: using computer that do 10^{15} Flops, targeting next-gen systems doing 10^{18} Flops, to solve problems with tens of billions (10^{12}) dofs

Iterative sparse linear solvers

A matrix is sparse when there are so many zeros (nonzeros are typically $\mathcal{O}(n)$) that it pays off to take advantage of them in the computer representation. *James Wilkinson*

Methods of choice: Look for an approximate solution by projection:

$$\begin{aligned} \mathbf{x}_m &\in \mathcal{K}_m(A, \mathbf{r}_0) \\ \mathbf{r}_m = \mathbf{b} - A\mathbf{x}_m &\perp \mathcal{K}_m(A, \mathbf{r}_0) \end{aligned}$$

$$\mathcal{K}_m(A, \mathbf{r}_0) = Span\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}$$

Krylov subspace (**growing with iteration until $\mathbf{x}_m$ is good enough**)

Iterative sparse linear solvers

A matrix is sparse when there are so many zeros (nonzeros are typically $\mathcal{O}(n)$) that it pays off to take advantage of them in the computer representation. *James Wilkinson*

Methods of choice: Look for an approximate solution by projection:

$$\begin{aligned} \mathbf{x}_m &\in \mathcal{K}_m(A, \mathbf{r}_0) \\ \mathbf{r}_m = \mathbf{b} - A\mathbf{x}_m &\perp \mathcal{K}_m(A, \mathbf{r}_0) \end{aligned}$$

$$\mathcal{K}_m(A, \mathbf{r}_0) = Span\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}$$

Krylov subspace (**growing with iteration until $\mathbf{x}_m$ is good enough**)
Conjugate Gradient (CG) for s.p.d. matrices (1952)

CG convergence

$$\frac{\|\mathbf{e}_k\|_A}{\|\mathbf{e}_0\|_A} \leq 2\left(\frac{a-1}{a+1}\right), \quad a = \sqrt{\kappa(A) = \lambda_{max}/\lambda_{min}}$$

$\mathbf{e}_k = \mathbf{x} - \mathbf{x}_k$ error at iteration k, λ eigenvalue of A

Preconditioning

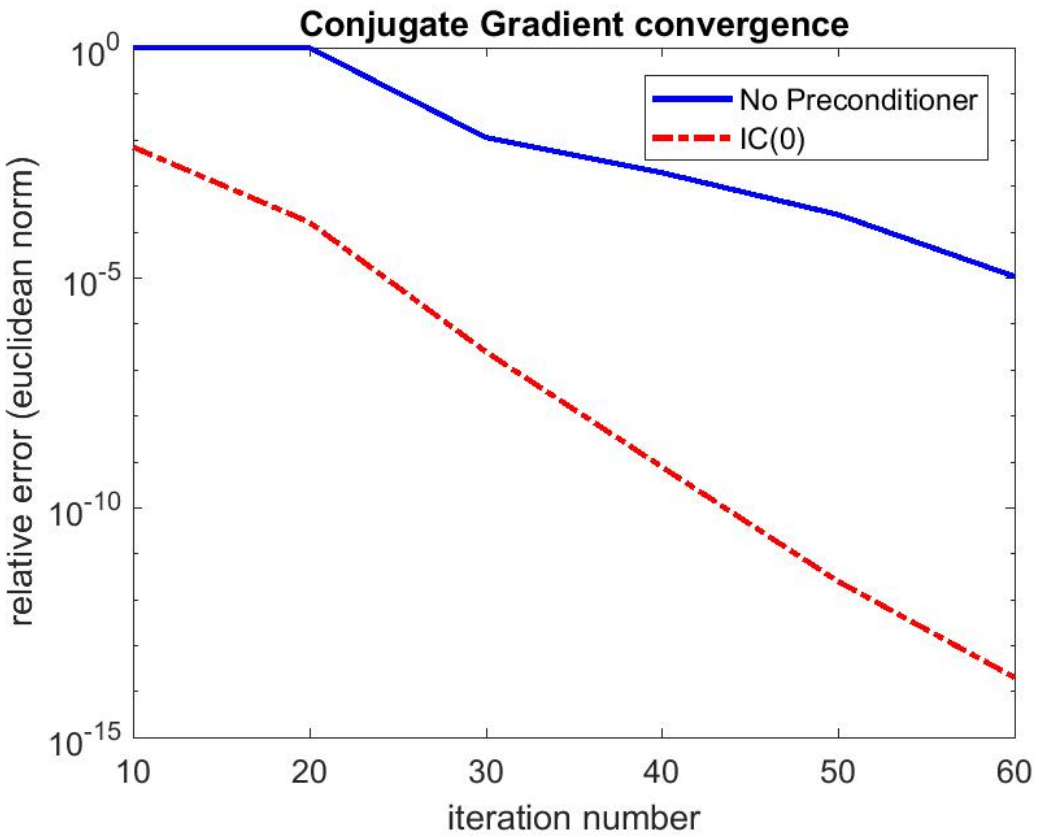
Solve the system:

$$BA\mathbf{x} = B\mathbf{b}$$

with $B \approx A^{-1}$
(left preconditioner)
such that:

$$\kappa(BA) << \kappa(A)$$

Solving 2D Poisson eq.
(2500 dofs, $\kappa(A) \approx 1.5 \times 10^3$)



IC(0): $B = (LL^T)^{-1}$ with L incompl. Cholesky factor, $\kappa(BA) \approx 2.2 \times 10^2$

The preconditioned Conjugate Gradient algorithm

1 Given $\mathbf{x}_0$ and set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$;
2 $\mathbf{z}_0 = B\mathbf{r}_0$;
3 $\mathbf{p}_0 = z_0$;
4 $\mathbf{w}_0 = A\mathbf{p}_0$;
5 **for** $i = 1, \ldots$ **do**
6 $\quad \alpha_{i-1} = \mathbf{r}_{i-1}^T \mathbf{z}_{i-1} / \mathbf{p}_{i-1}^T \mathbf{w}_{i-1}$;
7 $\quad \mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_{i-1} \mathbf{p}_{i-1}$;
8 $\quad \mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1} \mathbf{w}_{i-1}$;
9 $\quad$ **evaluate the stopping criterion**;
10 $\quad \mathbf{z}_i = B\mathbf{r}_i$;
11 $\quad \beta_i = \mathbf{r}_i^T \mathbf{z}_i / \mathbf{r}_{i-1}^T \mathbf{z}_{i-1}$;
12 $\quad \mathbf{p}_i = \mathbf{z}_i + \beta_i \mathbf{p}_{i-1}$;
13 $\quad \mathbf{w}_i = A\mathbf{p}_i$;
14 **end**

Building blocks

- preconditioner application
- *SpMV* operation involving the original matrix A
- *dot* products
- *axpy* operations

The preconditioned Conjugate Gradient algorithm

1 Given $\mathbf{x}_0$ and set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$;
2 $\mathbf{z}_0 = B\mathbf{r}_0$;
3 $\mathbf{p}_0 = z_0$;
4 $\mathbf{w}_0 = A\mathbf{p}_0$;
5 **for** $i = 1, \ldots$ **do**
6 $\quad \alpha_{i-1} = \mathbf{r}_{i-1}^T \mathbf{z}_{i-1} / \mathbf{p}_{i-1}^T \mathbf{w}_{i-1}$;
7 $\quad \mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_{i-1} \mathbf{p}_{i-1}$;
8 $\quad \mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1} \mathbf{w}_{i-1}$;
9 $\quad$ **evaluate the stopping criterion**;
10 $\quad \mathbf{z}_i = B\mathbf{r}_i$;
11 $\quad \beta_i = \mathbf{r}_i^T \mathbf{z}_i / \mathbf{r}_{i-1}^T \mathbf{z}_{i-1}$;
12 $\quad \mathbf{p}_i = \mathbf{z}_i + \beta_i \mathbf{p}_{i-1}$;
13 $\quad \mathbf{w}_i = A\mathbf{p}_i$;
14 **end**

Building blocks

- preconditioner application
- *SpMV* operation involving the original matrix A
- *dot* products
- *axpy* operations

Intrinsic performance limits

BLAS-1 (vector-vector) or BLAS-2 (sparse matrix-vector) operations

Compute intensity = Flops/Bytes = $\mathcal{O}(1)$

Memory (Communication) bound problems

Where we want to run[1]

	System	Cores	Rmax (PFlops)	**HPCG** (PFlops)
1	Frontier	8,730,112	1,102	14
2	Fugaku	7,630,848	442	16
3	LUMI	2,220,288	309	3.4
4	Leonardo	1,463,616	174	2.6
⋮	⋮	⋮	⋮	
24	Marconi-100	347,776	21	0.5
26	Piz Daint	387,872	21	0.5
⋮	⋮	⋮	⋮	
93	Juwels 1	114,480	6.18	0.075

Marconi 100 - Cineca

Piz Daint - CSCS

- Computers with thousands of CPU cores and GPU accelerators
- Hybrid form of parallelism/programming models: MPI, OpenMP, CUDA/OpenACC, ...

[1] TOP500 list, November 2022 – `https://www.top500.org`

Main issues and challenges

- the cost of data movement dominates the cost of floating-point arithmetic
- accelerators (GPUs, FPGAs, ...) can run at very high throughput exploiting high levels of data parallelism
- accelerators work very fast on low precision, floating-point arithmetic is available in hardware for fast AI
- minimizing energy consumption is important for sustainability of HPC

Main issues and challenges

- the cost of data movement dominates the cost of floating-point arithmetic needs to rethink numerical methods for reducing memory access and data communication among multiple processors
- accelerators (GPUs, FPGAs, ...) can run at very high throughput exploiting high levels of data parallelism
- accelerators work very fast on low precision, floating-point arithmetic is available in hardware for fast AI
- minimizing energy consumption is important for sustainability of HPC

Main issues and challenges

- the cost of data movement dominates the cost of floating-point arithmetic
- accelerators (GPUs, FPGAs, ...) can run at very high throughput exploiting high levels of data parallelism their efficient use often require to substitute more accurate methods with more parallel ones
- accelerators work very fast on low precision, floating-point arithmetic is available in hardware for fast AI
- minimizing energy consumption is important for sustainability of HPC

Main issues and challenges

- the cost of data movement dominates the cost of floating-point arithmetic
- accelerators (GPUs, FPGAs, ...) can run at very high throughput exploiting high levels of data parallelism
- accelerators work very fast on low precision, floating-point arithmetic is available in hardware for fast AI methods have to be proposed to exploit such computations within algorithms aiming for higher accuracy
- minimizing energy consumption is important for sustainability of HPC

Main issues and challenges

- the cost of data movement dominates the cost of floating-point arithmetic
- accelerators (GPUs, FPGAs, ...) can run at very high throughput exploiting high levels of data parallelism
- accelerators work very fast on low precision, floating-point arithmetic is available in hardware for fast AI
- minimizing energy consumption is important for sustainability of HPC basic guideline is reducing elapsed time of HPC applications and integrate energy consumption information into the algorithms

Main issues and challenges

- the cost of data movement dominates the cost of floating-point arithmetic
- accelerators (GPUs, FPGAs, ...) can run at very high throughput exploiting high levels of data parallelism
- accelerators work very fast on low precision, floating-point arithmetic is available in hardware for fast AI
- minimizing energy consumption is important for sustainability of HPC

the methods of "approximation mathematics" will have to be changed very radically in order to use . . .[a computer] sensibly and effectively - and to get into the position of being able to build and use still faster ones
(von Neumann, Letter to Maxwell Newman on 19 March 1946)

Main issues and challenges

- the cost of data movement dominates the cost of floating-point arithmetic
- accelerators (GPUs, FPGAs, ...) can run at very high throughput exploiting high levels of data parallelism
- accelerators work very fast on low precision, floating-point arithmetic is available in hardware for fast AI
- minimizing energy consumption is important for sustainability of HPC

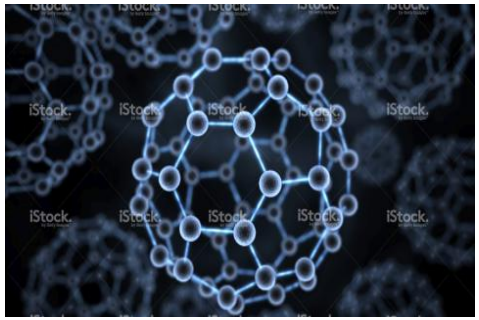
...the design of numerical algorithms and mathematical software is an interdisciplinary scientific topic with many features of a translational science which requires a continuous feedback from the applications to the basic research
(J. Dongarra, Journal of Computational Science, 2021)

EoCoE project

Energy oriented Center of Excellence: toward exascale for energy

applying cutting-edge computational methods to accelerate the transition to the production, storage and management of clean, decarbonized energy

Wind



Materials

Water

Fusion

Main aim

prepare selected applications to face the exascale challenge

EoCoE target applications

Wind Models

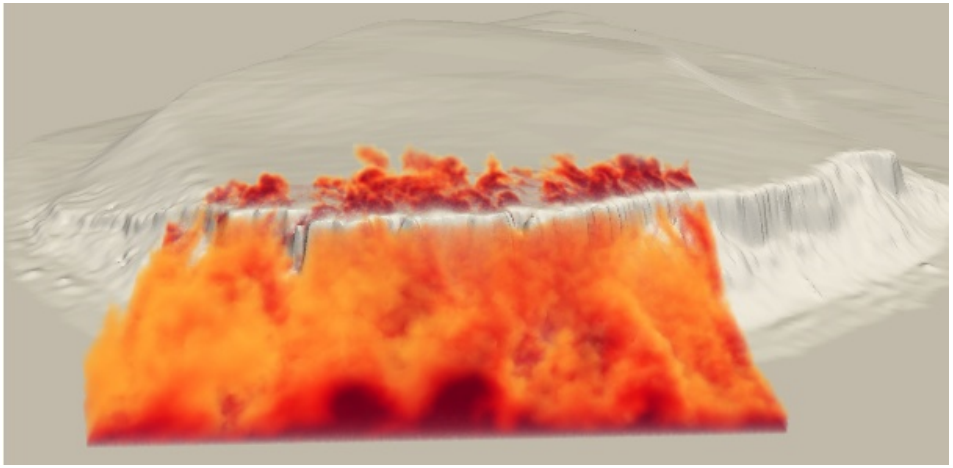
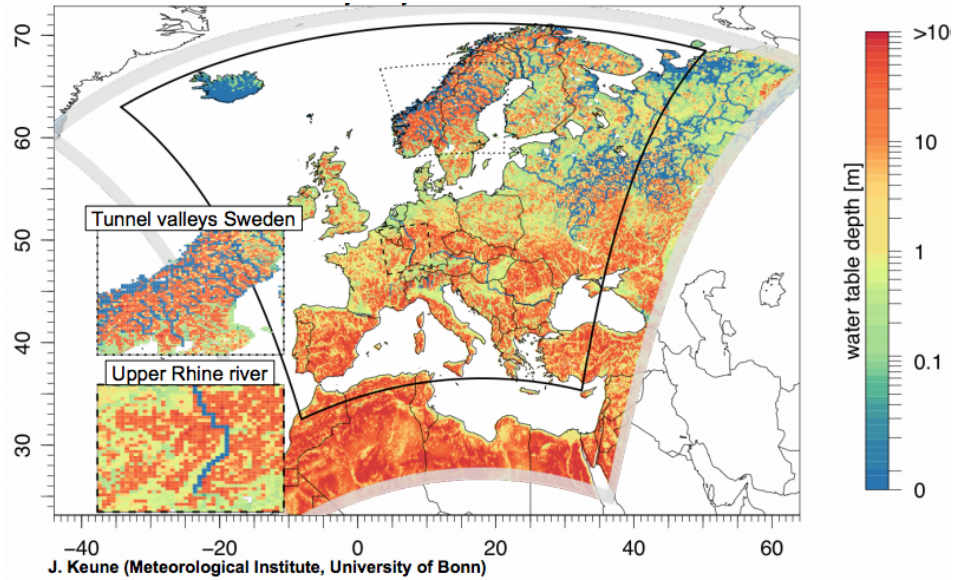


Image credits H. Owen and G. Marin, Barcelona Supercomputing Centre

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

Regional Hydrological Models



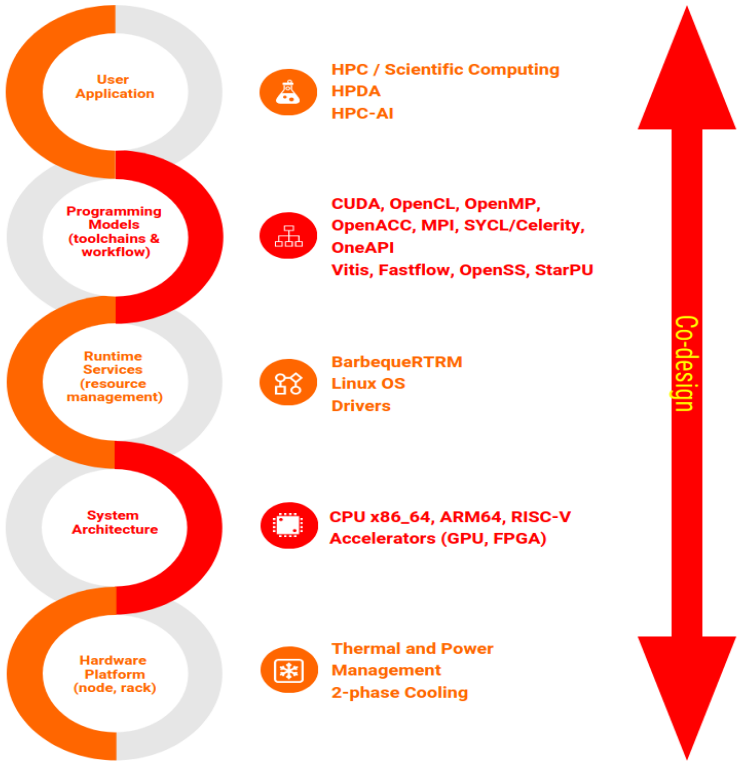
- Darcy equation,
- Richards equation,
- Equations for overland flow
- ...

Target dofs: $n > 10^{12}$, Computing processors: $np \approx 10^6$

TEXTAROSSA project

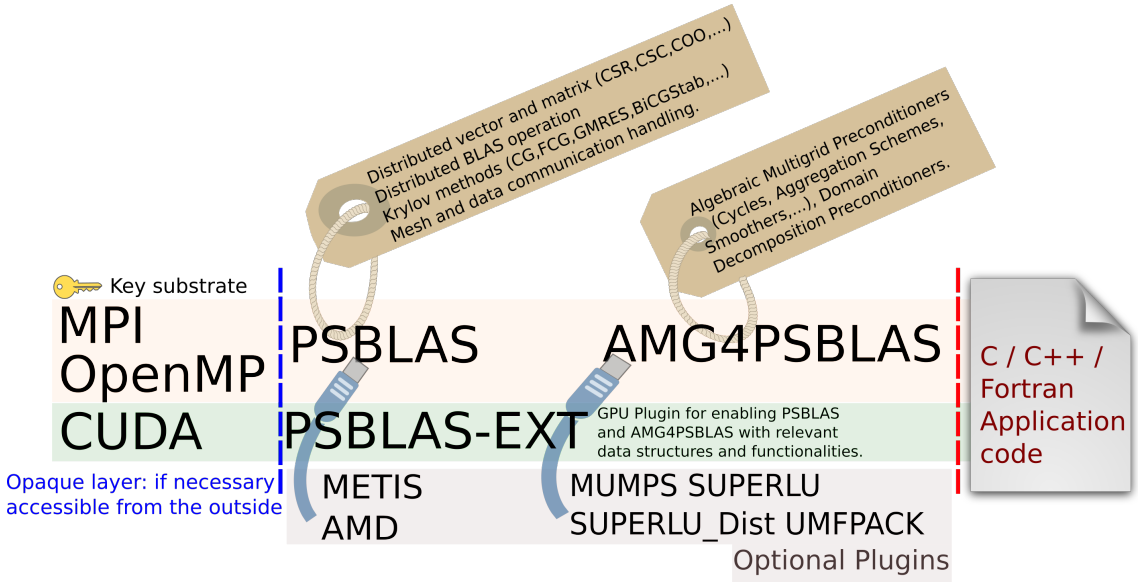
Towards EXtreme scale Technologies and Accelerators for euROhpc hw/Sw Supercomputing Applications for exascale

developing new software tools for high-performance and high-energy efficiency on near-future exascale computing systems by multi-directional co-design approach

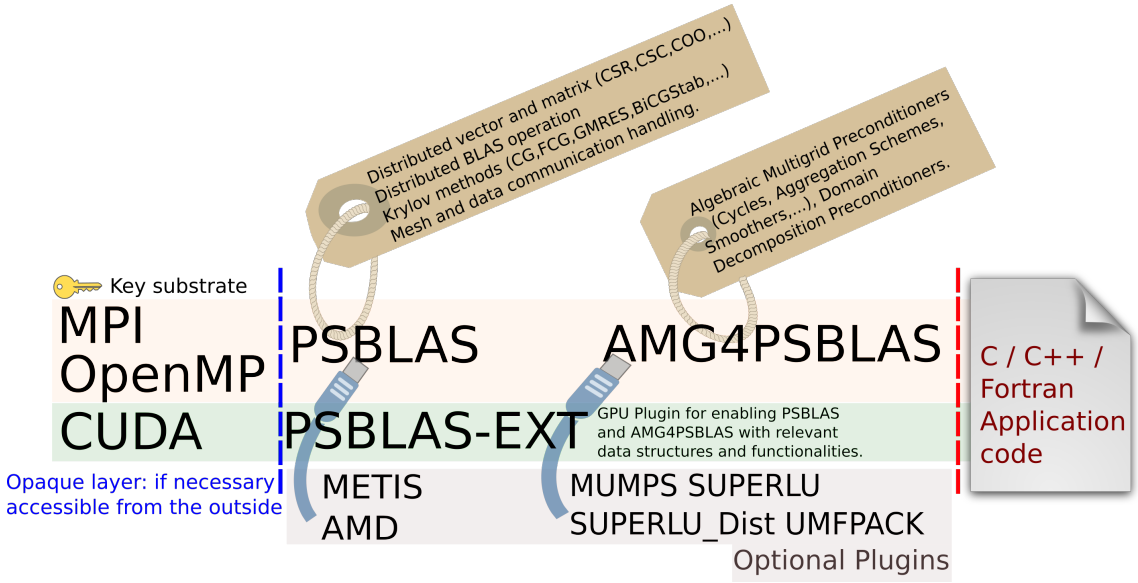


Our contribution: performance/power efficient MathLib

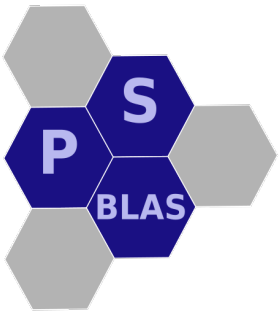
Parallel Sparse Computation toolkit (psctoolkit.github.io)

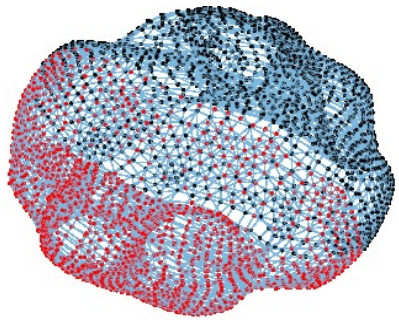


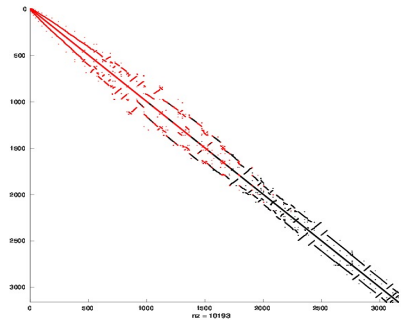
Parallel Sparse Computation toolkit (psctoolkit.github.io)



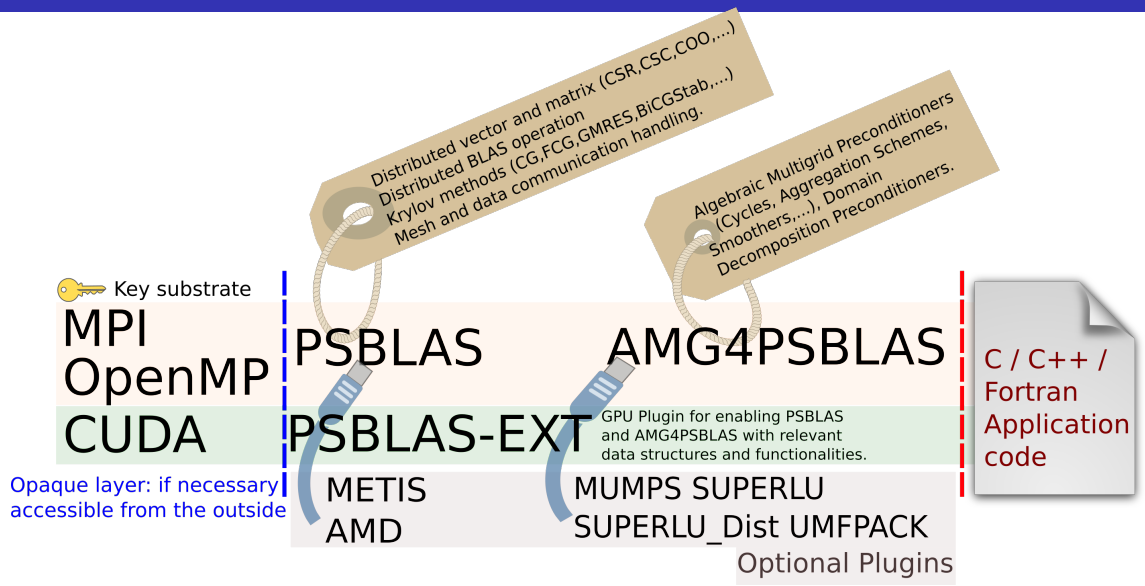
SPMD programming model; parallel sparse BLAS-1/2/3; Krylov solvers; algebraic interface with support for mesh handling and partitioning



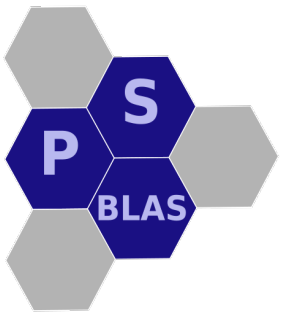


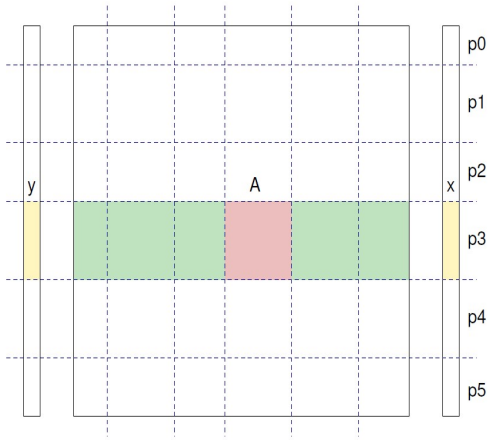


Parallel Sparse Computation toolkit (psctoolkit.github.io)

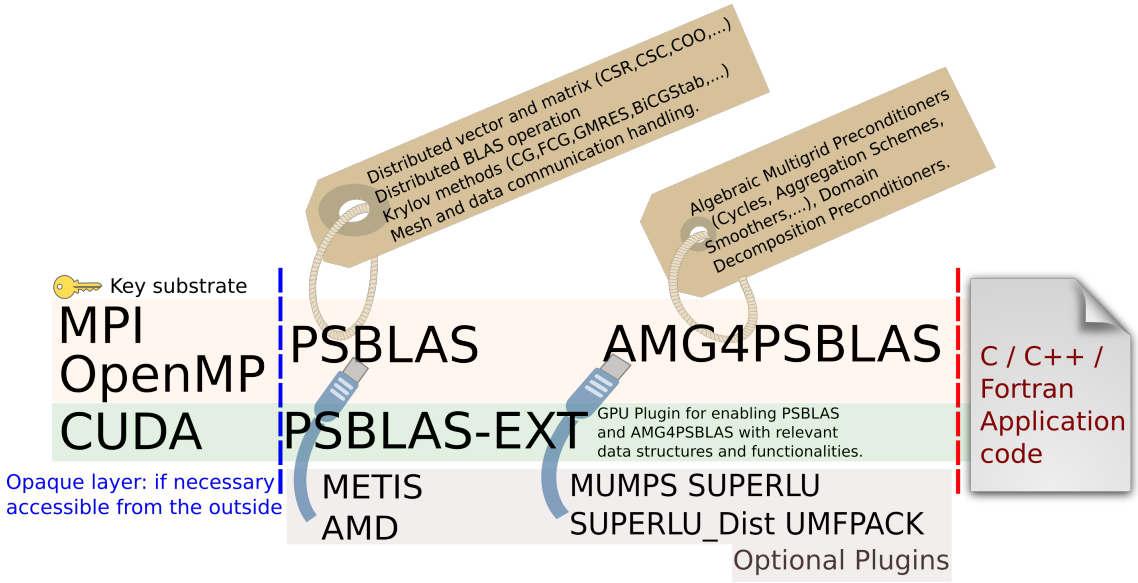


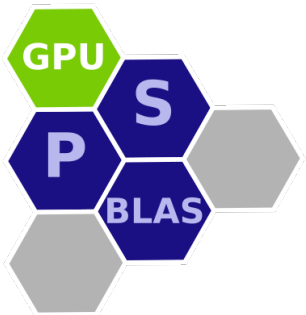
effective handling of large index spaces
and of halo data exchange
(pink area is local, green area is halo)





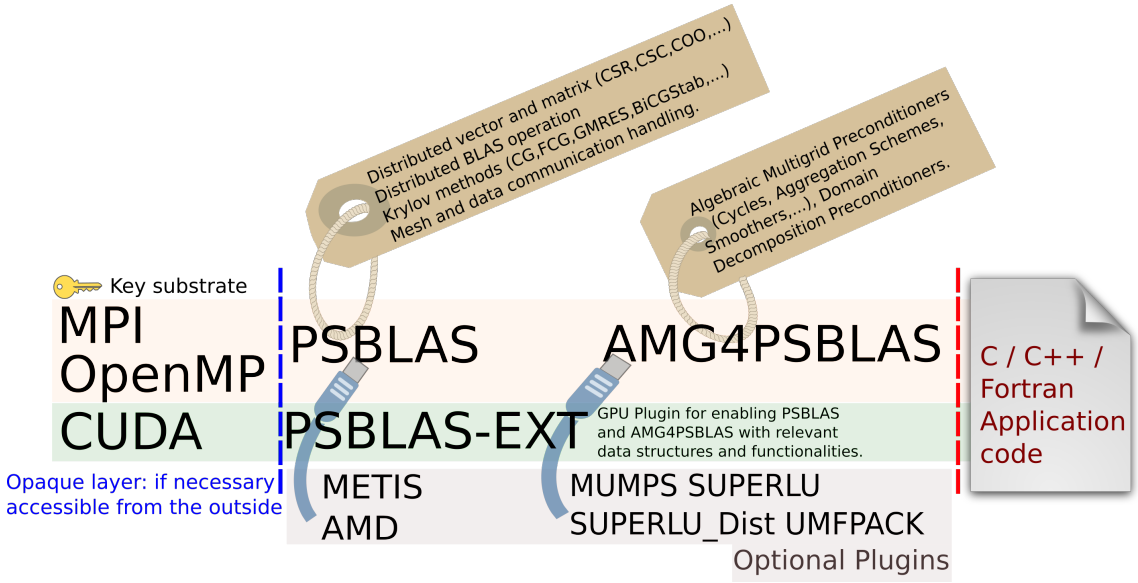
Parallel Sparse Computation toolkit (psctoolkit.github.io)

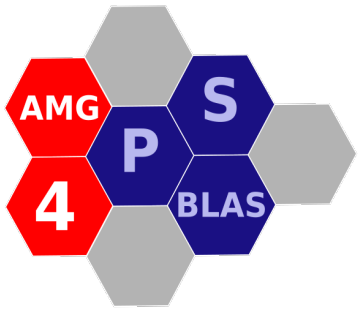




Additional matrix storage formats, interfaces to two external libraries for sparse BLAS-1/2 on NVIDIA GPUs and on multi-core CPUs

Parallel Sparse Computation toolkit (psctoolkit.github.io)





a package of parallel algebraic multigrid preconditioners, specifically designed and implemented for extreme-scale computations

MultiGrid methods

Given $A \in \mathbb{R}^{n\times n}$ s.p.d., apply B to precondition the CG solver:

if $(k \neq nlev)$ then

$x^k = x^k + (M^k)^{-1}\left(b^k - A^k x^k\right)$

$b^{k+1} = \left(P_k^{k+1}\right)^T \left(b^k - A^k x^k\right)$

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

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

$x^k = x^k + (M^k)^{-T}\left(b^k - A^k x^k\right)$

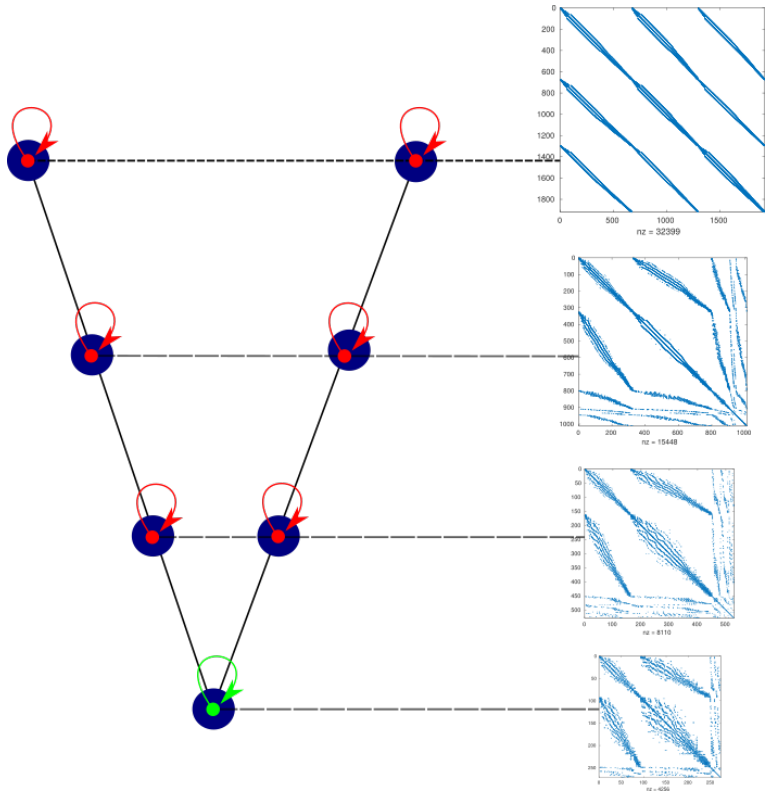
else

$x^k = \left(A^k\right)^{-1} b^k$

endif

return x^k

end



Smoother

$M^k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$

"High frequencies"

Prolongator

$P_k^{k+1} : \mathbb{R}^{n_{k+1}} \to \mathbb{R}^{n_k}$

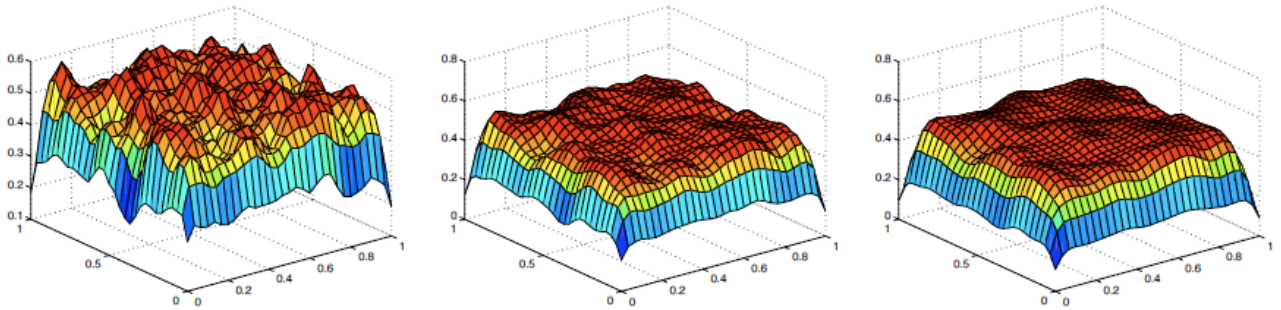
"Low frequencies"

Complementarity of Smoother and Prolongator

Algebraic MultiGrid (AMG) methods

Brandt, McCormick and Ruge (1984)

Algebraic MultiGrid methods do not explicitly use the (eventual) problem geometry but rely only on matrix entries to generate coarse-grids by using characterizations of *algebraic smoothness*



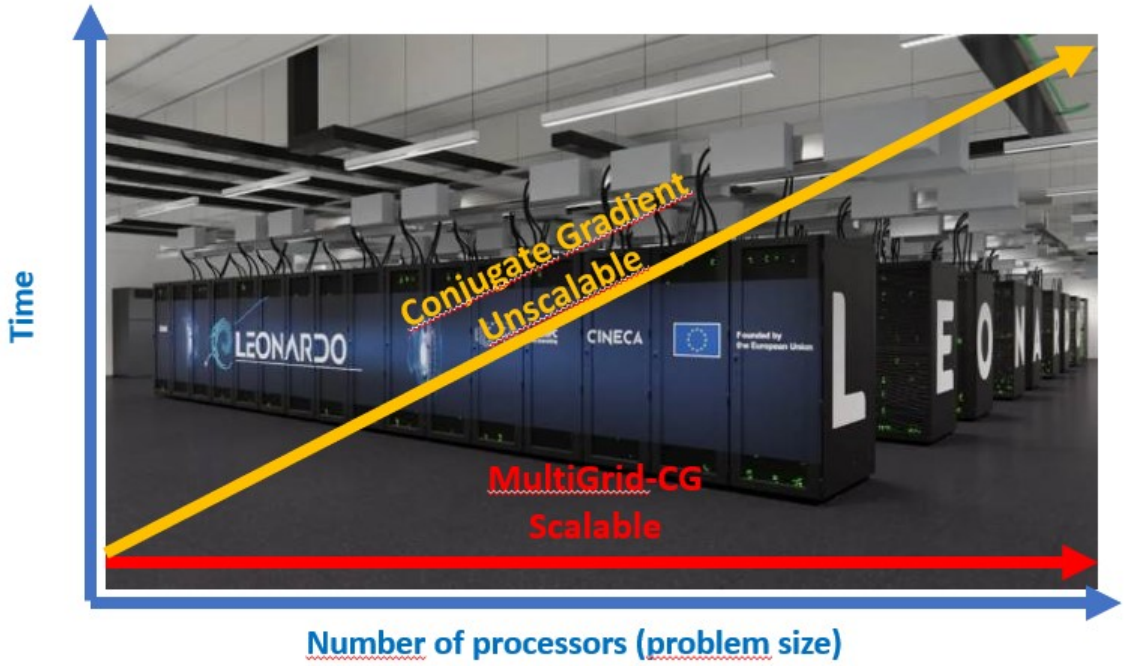
Key issue

errors not reduced by the (chosen) smoother (algebraic smoothness):

$$(Aw)_i = r_i \approx 0 \Longrightarrow w_{i+1} \approx w_i$$

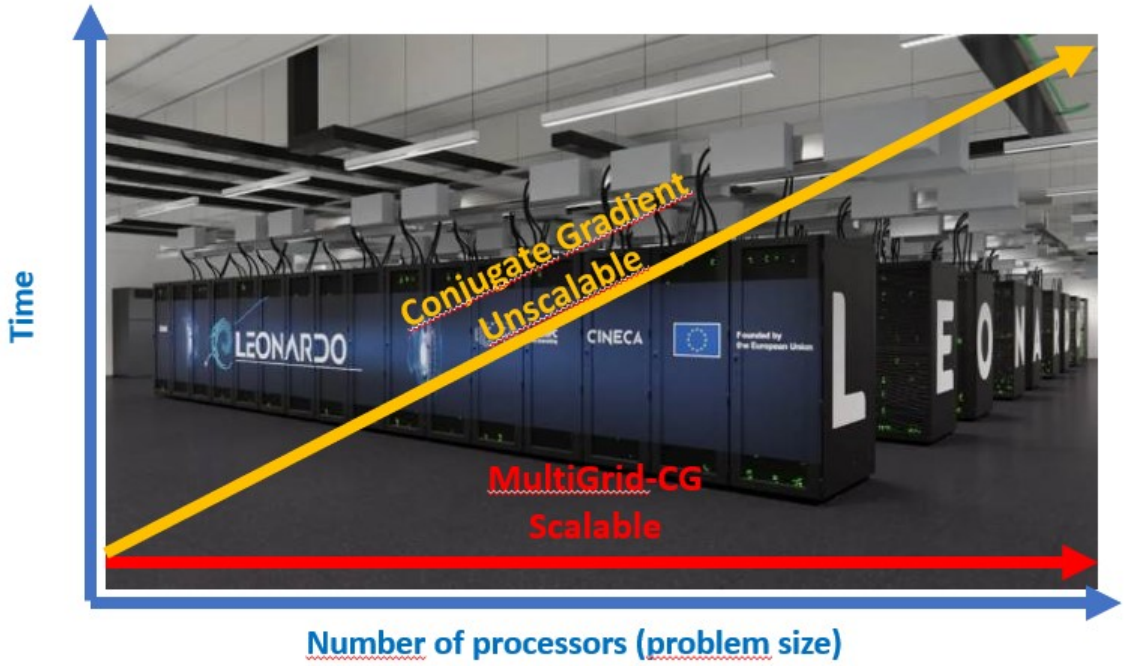
have to be well represented on the coarse grid and well interpolated back $\mathbf{w} = (w_i) \in \mathcal{R}ange(P^k_{k+1})$

Scalable (AMG) preconditioners



AMG can be optimal ($\mathcal{O}(n)$ flops) and hence have good scalability potential

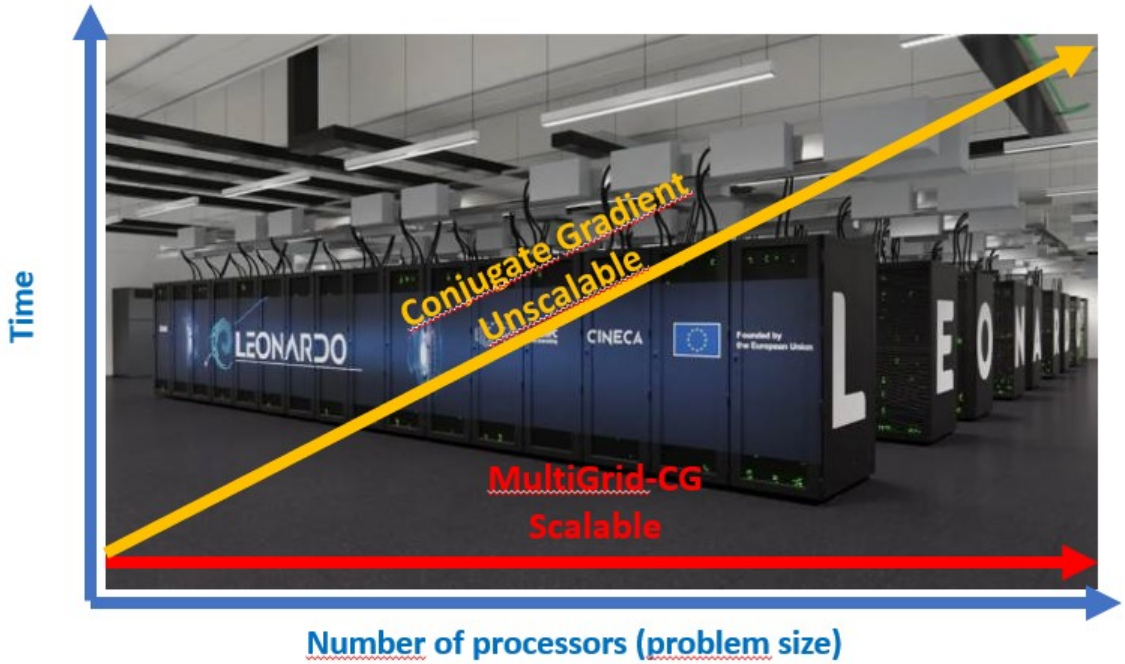
Scalable (AMG) preconditioners



AMG can be optimal ($\mathcal{O}(n)$ flops) and hence have good scalability potential

Optimal complexity is not sufficient in parallel!

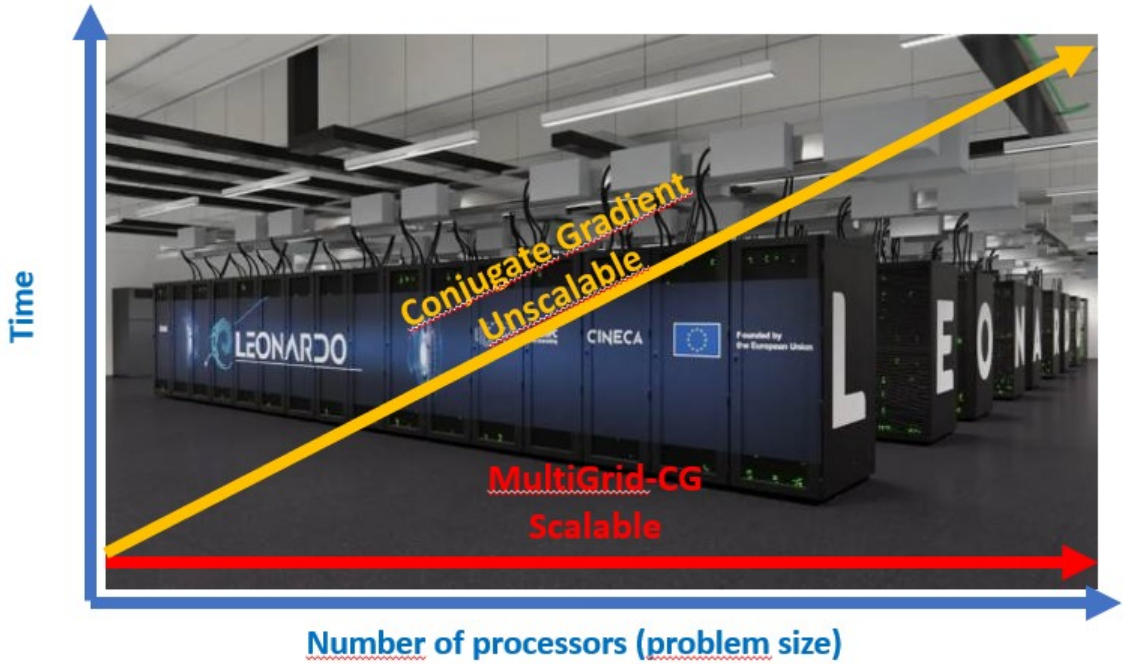
Scalable (AMG) preconditioners



AMG can be optimal ($\mathcal{O}(n)$ flops) and hence have good scalability potential
Optimal complexity is not sufficient in parallel!

- $\max_i \lambda_i(BA) \approx 1$ being independent of n (algorithmic scalability)

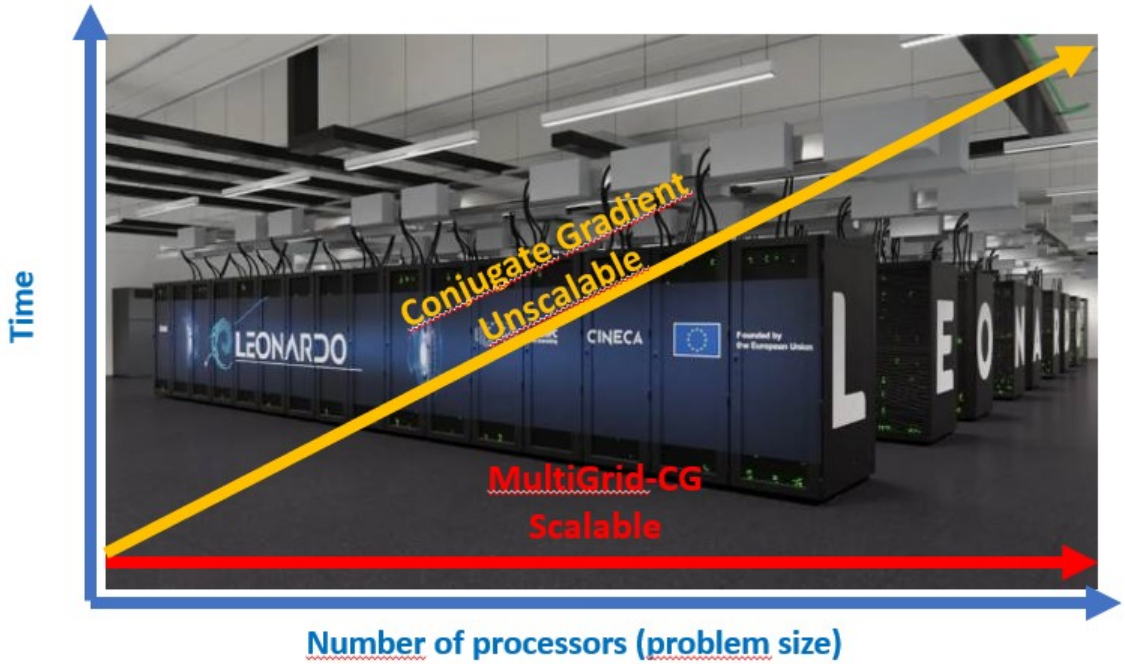
Scalable (AMG) preconditioners



AMG can be optimal ($\mathcal{O}(n)$ flops) and hence have good scalability potential
Optimal complexity is not sufficient in parallel!

- $\max_i \lambda_i(BA) \approx 1$ being independent of n (algorithmic scalability)
  true only for Laplacian and surroundings!

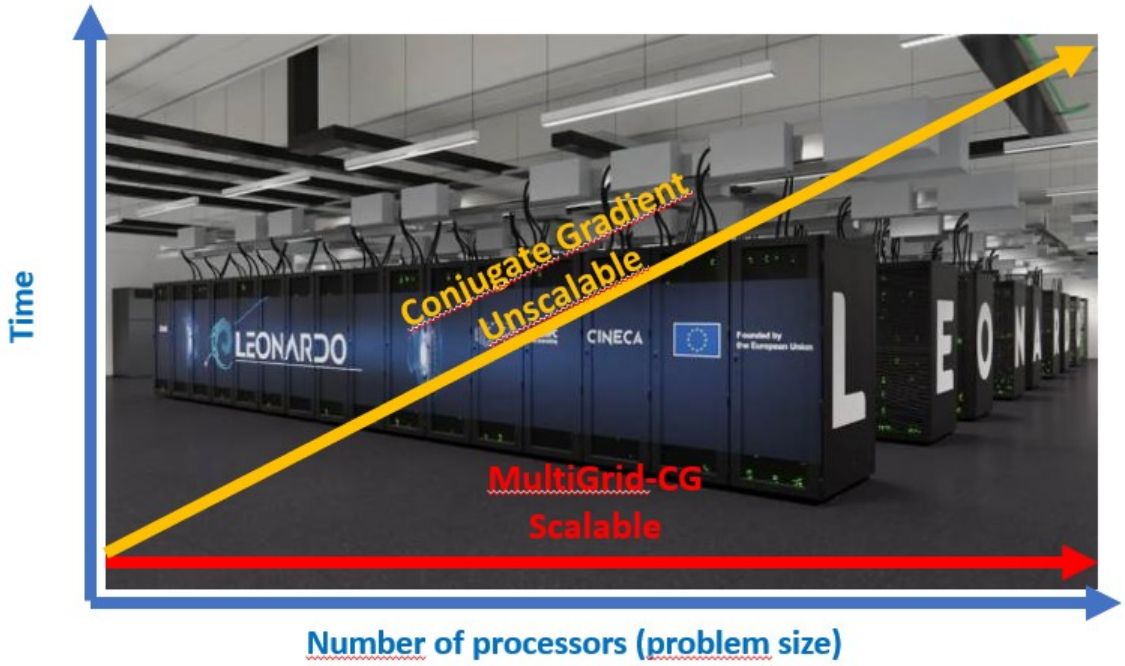
Scalable (AMG) preconditioners



AMG can be optimal ($\mathcal{O}(n)$ flops) and hence have good scalability potential
Optimal complexity is not sufficient in parallel!

- in a massively parallel computer, B should be composed of local actions (implementation scalability)

Scalable (AMG) preconditioners



AMG can be optimal ($\mathcal{O}(n)$ flops) and hence have good scalability potential
Optimal complexity is not sufficient in parallel!

- in a massively parallel computer, B should be composed of local actions (implementation scalability)
  limited by basic BLAS-2 (Sparse Matrix-Vector product) operations and reduced parallelism on coarser levels

AMG setup

Recursive application of a two-grid scheme

- setup of a convergent iterative solver M (the smoother)
- setup of a coarse vector space $\mathcal{R}^{n_c}$ from $\mathcal{R}^n$
- build the prolongation P from A
- compute coarse grid matrix $A_c = P^T AP$ (triple-matrix Galerkin product)

AMG setup

Recursive application of a two-grid scheme

- setup of a convergent iterative solver M (the smoother)
- setup of a coarse vector space $\mathcal{R}^{n_c}$ from $\mathcal{R}^n$
- build the prolongation P from A
- compute coarse grid matrix $A_c = P^T A P$ (triple-matrix Galerkin product)

Our recipies: AMG based on aggregation of dofs

Group the dofs into disjoint sets of aggregates G_j; each aggregate G_j corresponds to 1 coarse dof

Associated prolongation:

$$P := P_{ij} = \begin{cases} w_i & \text{if } i \in G_j \\ 0 & \text{otherwise} \end{cases}$$

$$i = 1, \ldots, n, \; j = 1, \ldots, n_c,$$

or smoothed version of P

(Vaněk, Mandel and Brezina 1996).

Parallel AMG setup: matching-based coupled aggregation

AMG based on weighted graph matching

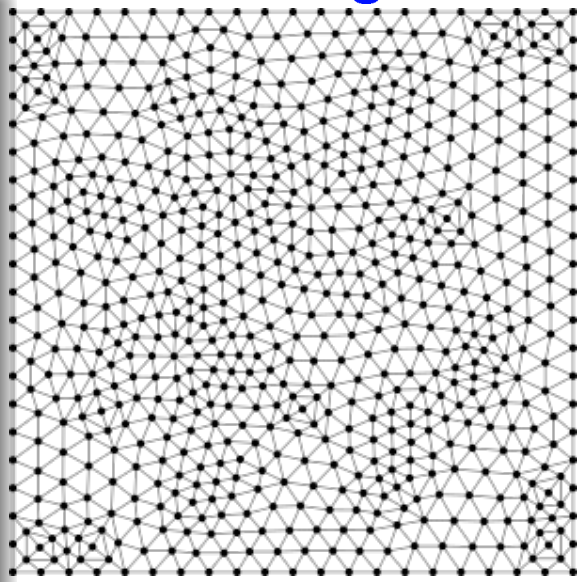
Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight (smooth) vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{A}$:

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

- 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 $e_{i \mapsto j}$.

P. D'Ambra and P. S. Vassilevski 2013

CMATCH algorithm



Parallel AMG setup: matching-based coupled aggregation

AMG based on weighted graph matching

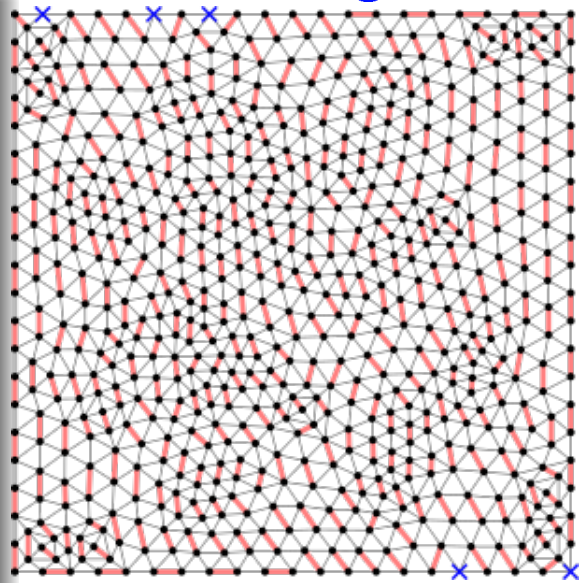
Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight (smooth) vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{A}$:

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

- 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 $e_{i \mapsto j}$.

P. D'Ambra and P. S. Vassilevski 2013

CMATCH algorithm



We 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 $i \neq j$, and unmatched vertices, i.e., n_s singletons G_i

Parallel AMG setup: matching-based coupled aggregation

AMG based on weighted graph matching

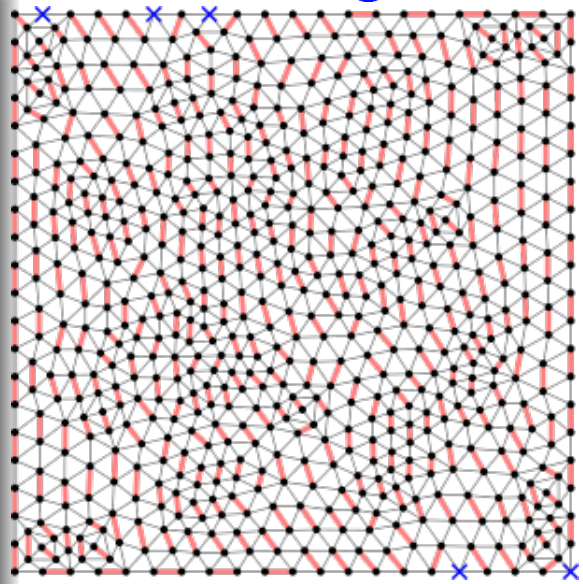
Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight (smooth) vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{A}$:

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

- 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 $e_{i \mapsto j}$.

P. D'Ambra and P. S. Vassilevski 2013

CMATCH algorithm



To increase coarsening ratio we can perform more than one sweep of matching per level

Parallel AMG setup: matching-based coupled aggregation

AMG based on weighted graph matching

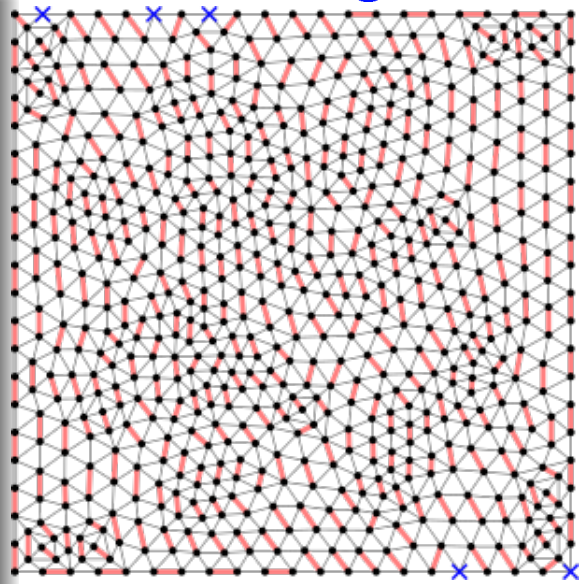
Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight (smooth) vector $\mathbf{w}$ we consider the weighted version of G obtained by considering the weight matrix $\hat{A}$:

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

- 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 $e_{i \mapsto j}$.

P. D'Ambra and P. S. Vassilevski 2013

CMATCH algorithm



To increase regularity of P_{l+1}^l we can consider a smoothed prolongator by applying one step of Jacobi method

Parallel AMG setup: matching-based coupled aggregation

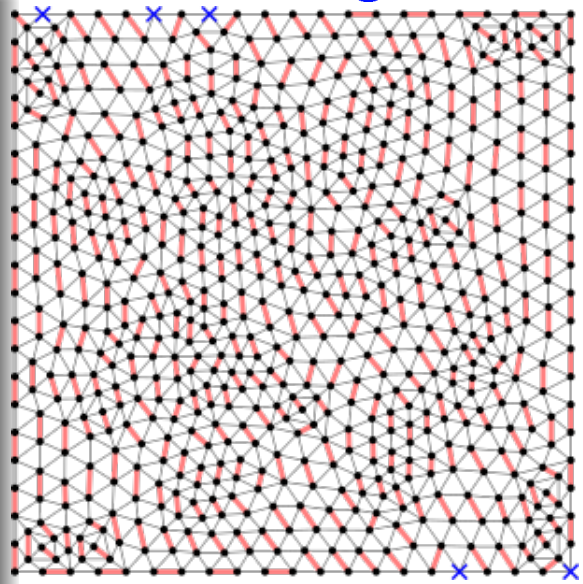
AMG based on weighted graph matching

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 $e_{i \mapsto j}$.

CMATCH algorithm



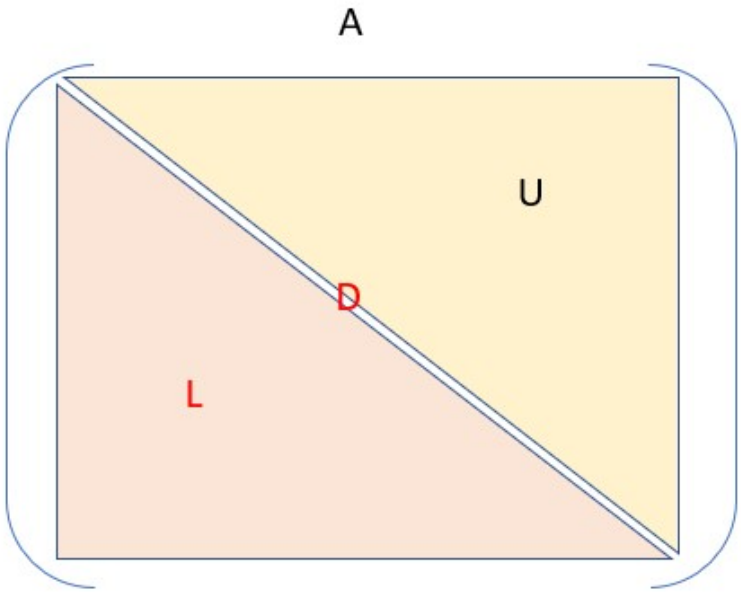
P. D'Ambra and P. S. Vassilevski 2013

Main building block: parallel approximated matching

sub-optimal algorithms with quality guarantee of the computed matching and linear-time $\mathcal{O}(nnz)$ complexity. Available software: MatchBox-P by Halappanavar et al.

Highly parallel smoothers: our recipies

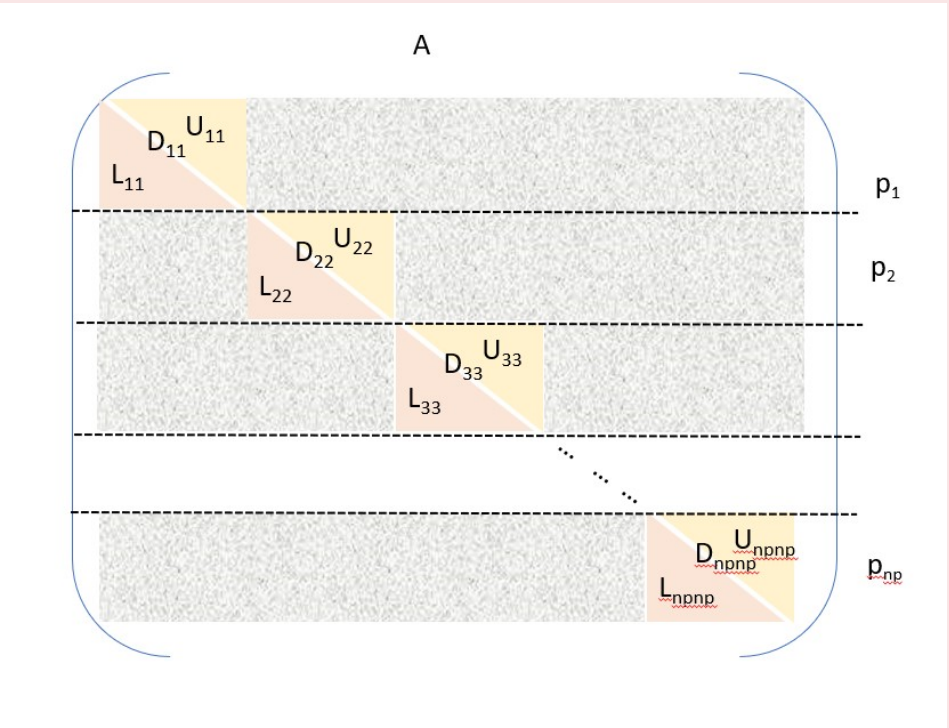
Gauss-Seidel (GS): $A = L + D + U$, where
$D = \mathrm{diag}(A)$, $L = \mathrm{tril}(A)$ and $U = \mathrm{triu}(A)$
the smoother is $M = (L + D)^{-1}U$,
It is intrinsically sequential!



Highly parallel smoothers: our recipies

Inexact block-Jacobi (HGS/weighted-Jacobi)

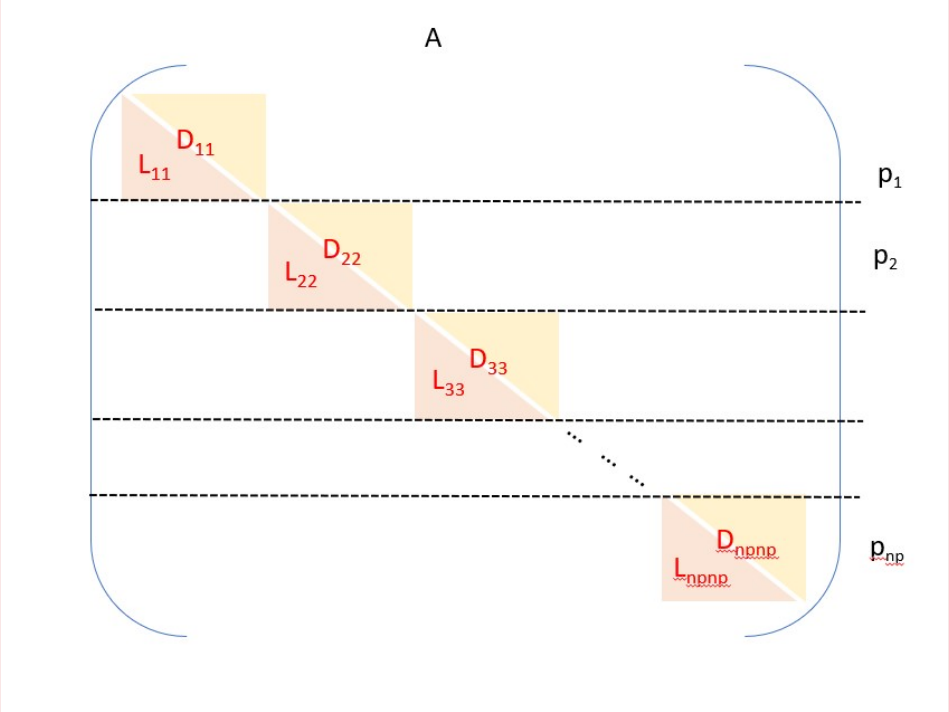
On process p, $A_{pp} = L_{pp} + D_{pp} + U_{pp}$
where $D_{pp} = \mathrm{diag}(A_{pp})$, $L_{pp} = \mathrm{tril}(A_{pp})$, $U_{pp} = \mathrm{triu}(A_{pp})$



Highly parallel smoothers: our recipies

Inexact block-Jacobi (HGS/weighted-Jacobi)

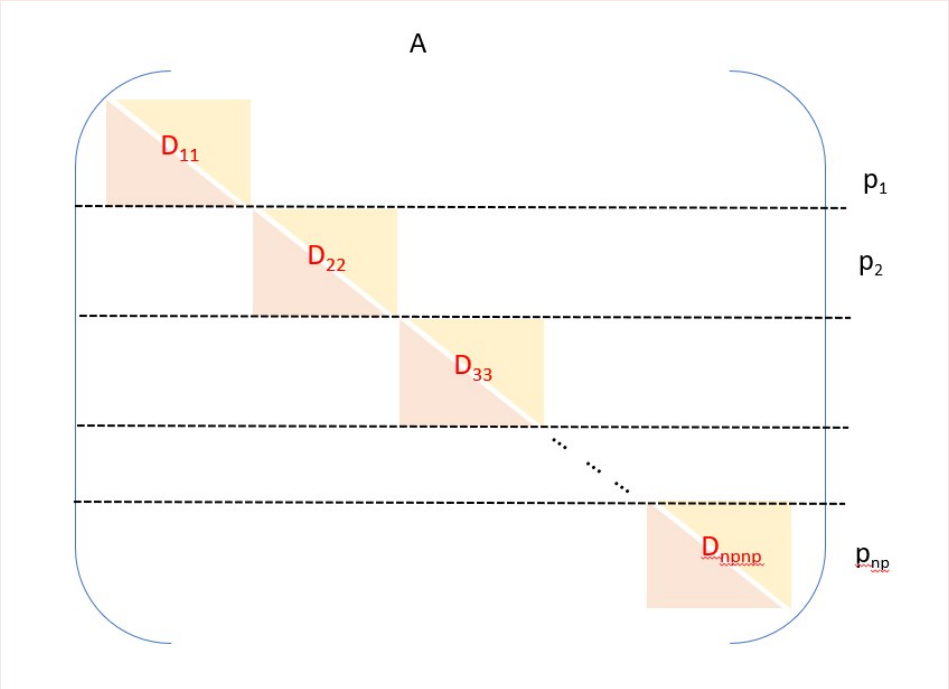
On process p, $A_{pp} = L_{pp} + D_{pp} + U_{pp}$
where $D_{pp} = \mathrm{diag}(A_{pp})$, $L_{pp} = \mathrm{tril}(A_{pp})$, $U_{pp} = \mathrm{triu}(A_{pp})$
HGS: the smoother is
$M = \mathrm{blockdiag}(M_{pp})$, with $M_{pp} = (L_{pp} + D_{pp})^{-1} U_{pp}$



Highly parallel smoothers: our recipies

Inexact block-Jacobi (HGS/weighted-Jacobi)

On process p, $A_{pp} = L_{pp} + D_{pp} + U_{pp}$
where $D_{pp} = \mathrm{diag}(A_{pp})$, $L_{pp} = \mathrm{tril}(A_{pp})$, $U_{pp} = \mathrm{triu}(A_{pp})$
weighted Jacobi: the smoother is
$M = \mathrm{blockdiag}(D_{pp}^{-1})$
worst smoothing properties but very suitable for GPUs



Test case: Poisson equation (as in HPCG)

$$-\Delta u = 1 \quad \text{on unit cube, with DBC}$$

- 7-point finite-difference discretization
- cartesian grid with uniform refinement along the coordinates for increasing mesh size

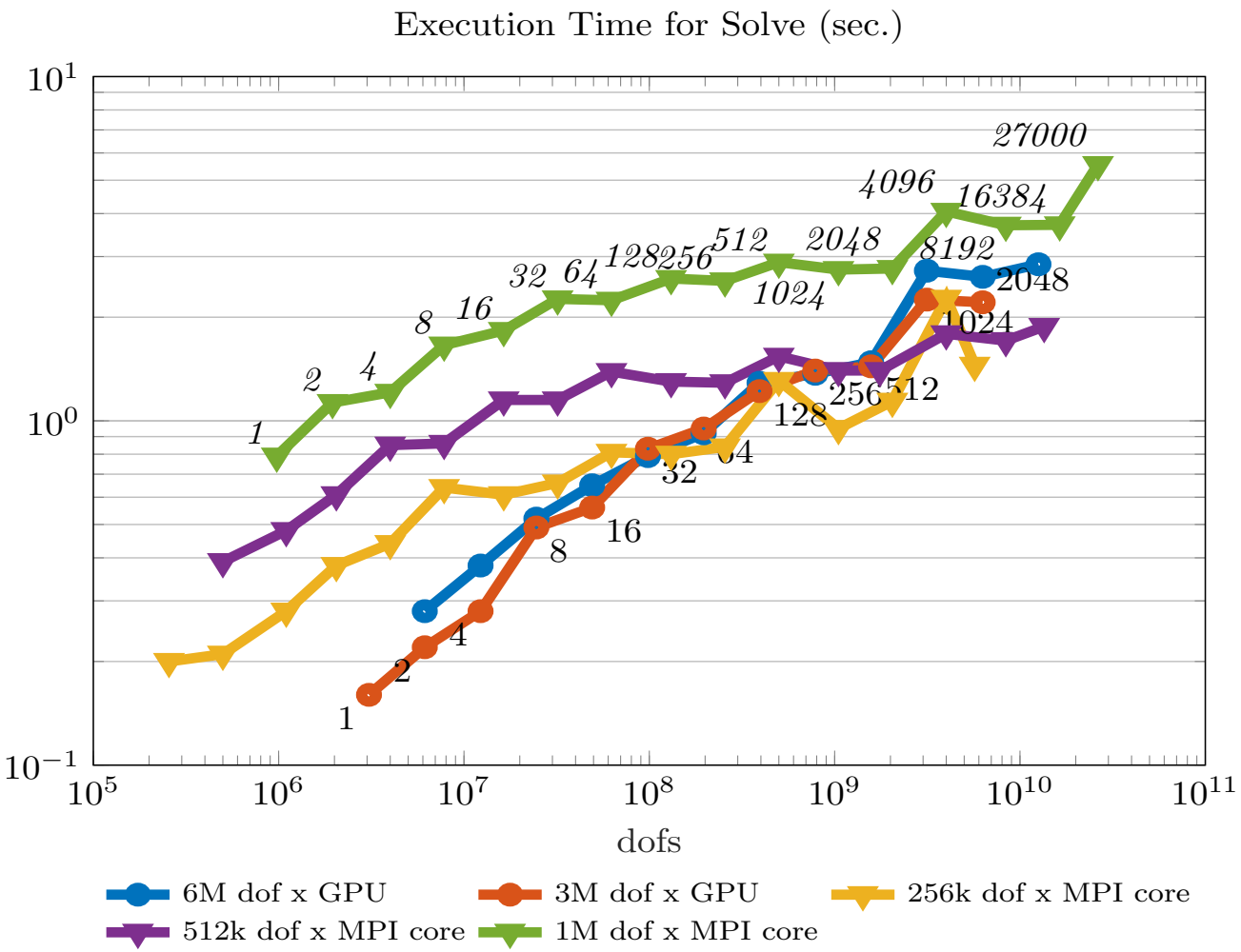
Solver/preconditioner settings

- AMG as preconditioner of CG, stopped when $\|\mathbf{r}^k\|_2/\|\mathbf{b}\|_2 \leq 10^{-6}$, or $itmax = 500$

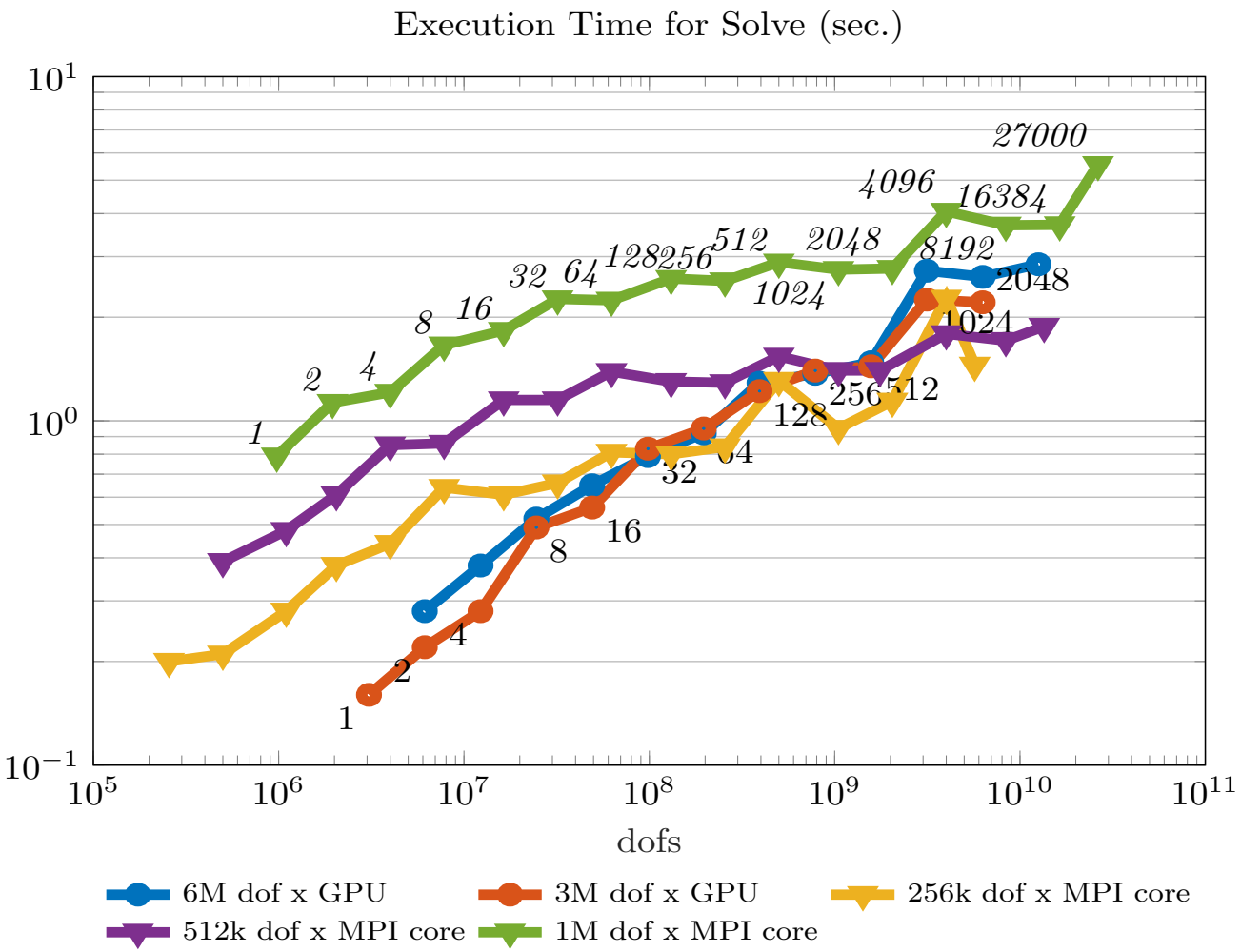
  VSCMATCH V-cycle, CMATCH building aggregates of max size 8, smoothed prolongators
- coarsest matrix size $n_c \leq 200np$, with np number of cores
- 1 sweep of forward/backward Hybrid Gauss-Seidel smoother (4 sweeps of weighted-Jacobi on GPU), parallel PCG coupled with Block-Jacobi+ILU(0) at the coarsest level.

Platform: Piz Daint, Cray Model XC40/Cray XC50 with 5704 hybrid compute nodes (Intel Xeon E5-2690 v3 with Nvidia Tesla P100)

Results at extreme scale: MPI vs hybrid MPI-CUDA



Results at extreme scale: MPI vs hybrid MPI-CUDA

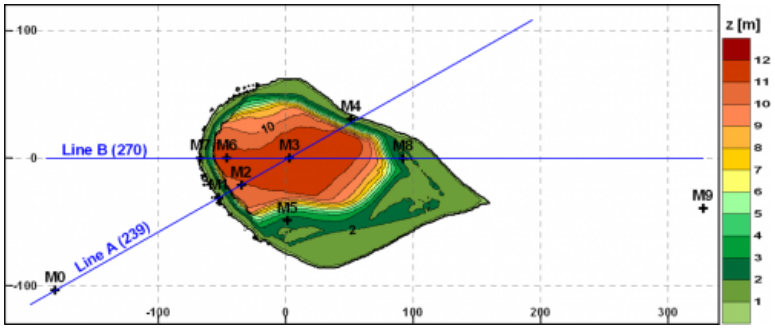


Performance/Power efficiency

the hybrid approach permits savings in solve time and energy consumption

A CFD application inside Alya

Joint work with
Herbert Owen
Barcelona Super Computing Center

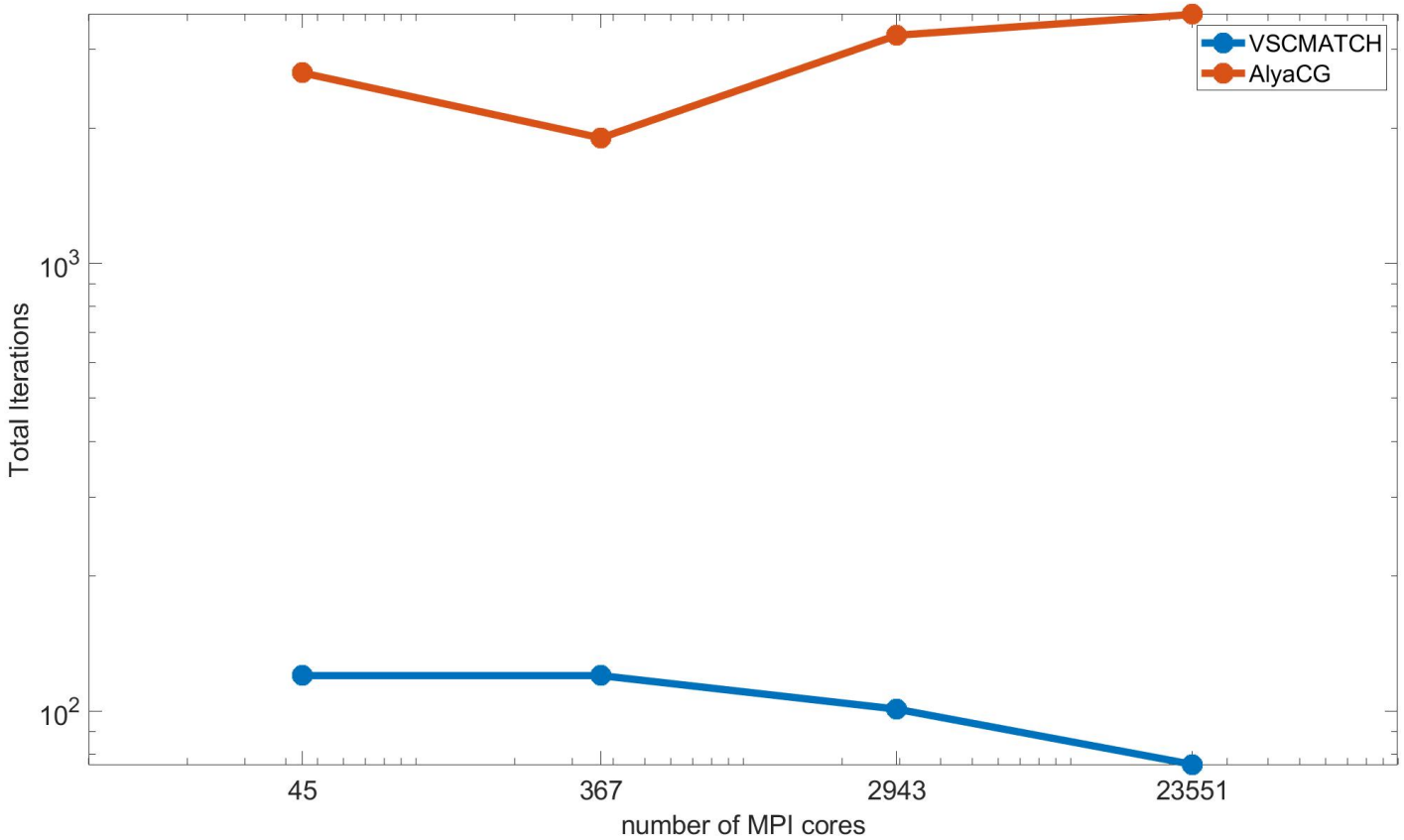


Bolund is an isolated hill situated in Roskilde Fjord, Denmark. An almost vertical escarpment in the prevailing W-SW sector ensures flow separation in the windward edge resulting in a complex flow field.

- **Model**: 3D incompressible unsteady Navier-Stokes equations for Large Eddy Simulations of turbulent flows $Re_\tau = 10^7$
- **Discretization**: low-dissipation mixed FEM (linear FEM both for velocity and pressure)
- **Time-Stepping**: non-incremental fractional-step for pressure, explicit fourth order Runge-Kutta method for velocity

Bolund test case - weak scaling on Juwels (JSC) - pressure equation

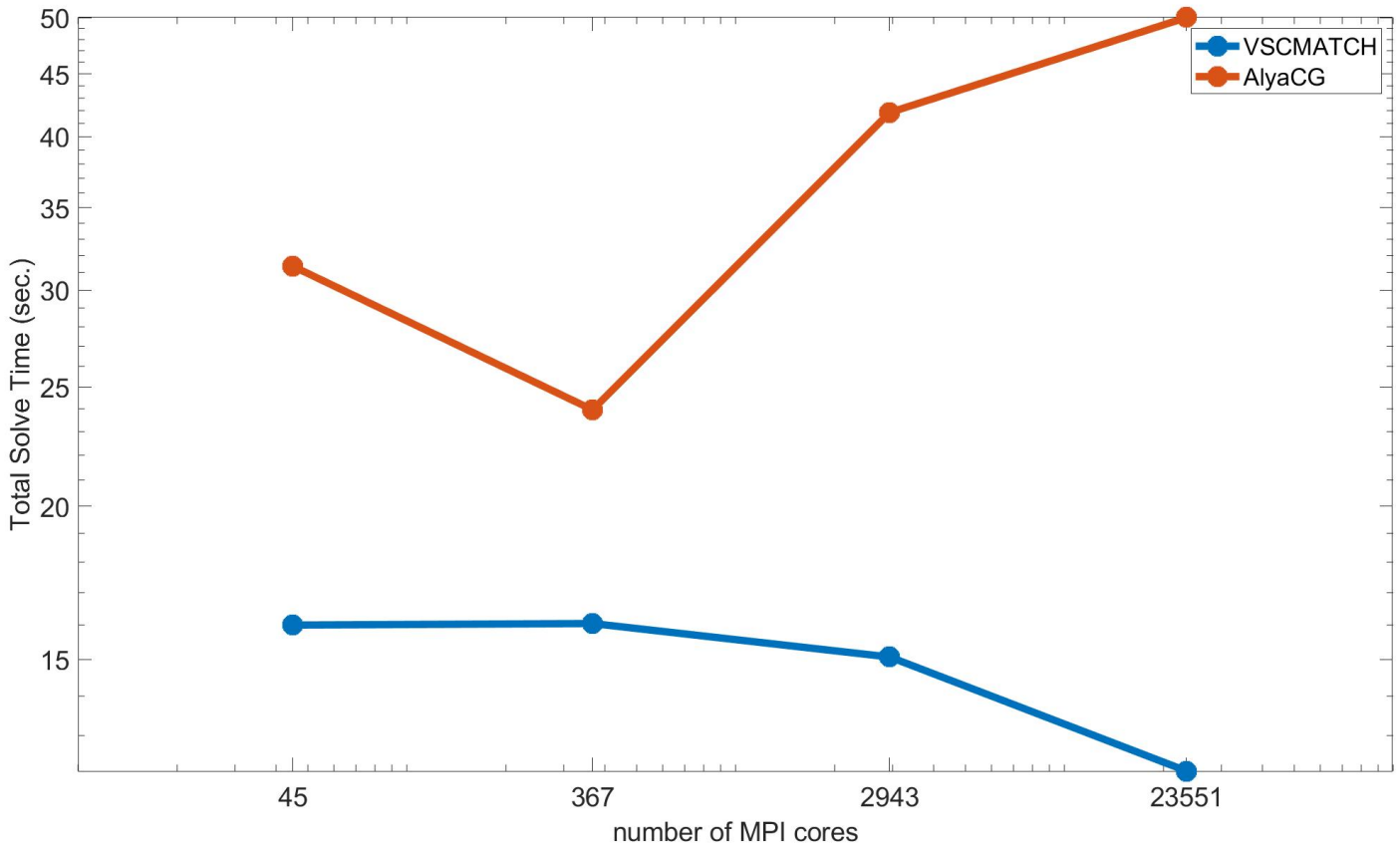
fixed size problem per CPU core $\approx 10^5$ dofs up to 2.9×10^9 dofs
20 time steps in the fully development flow phase



PSCToolkit solver largely reduces the total number of iterations

Bolund test case - weak scaling on Juwels (JSC) - pressure equation

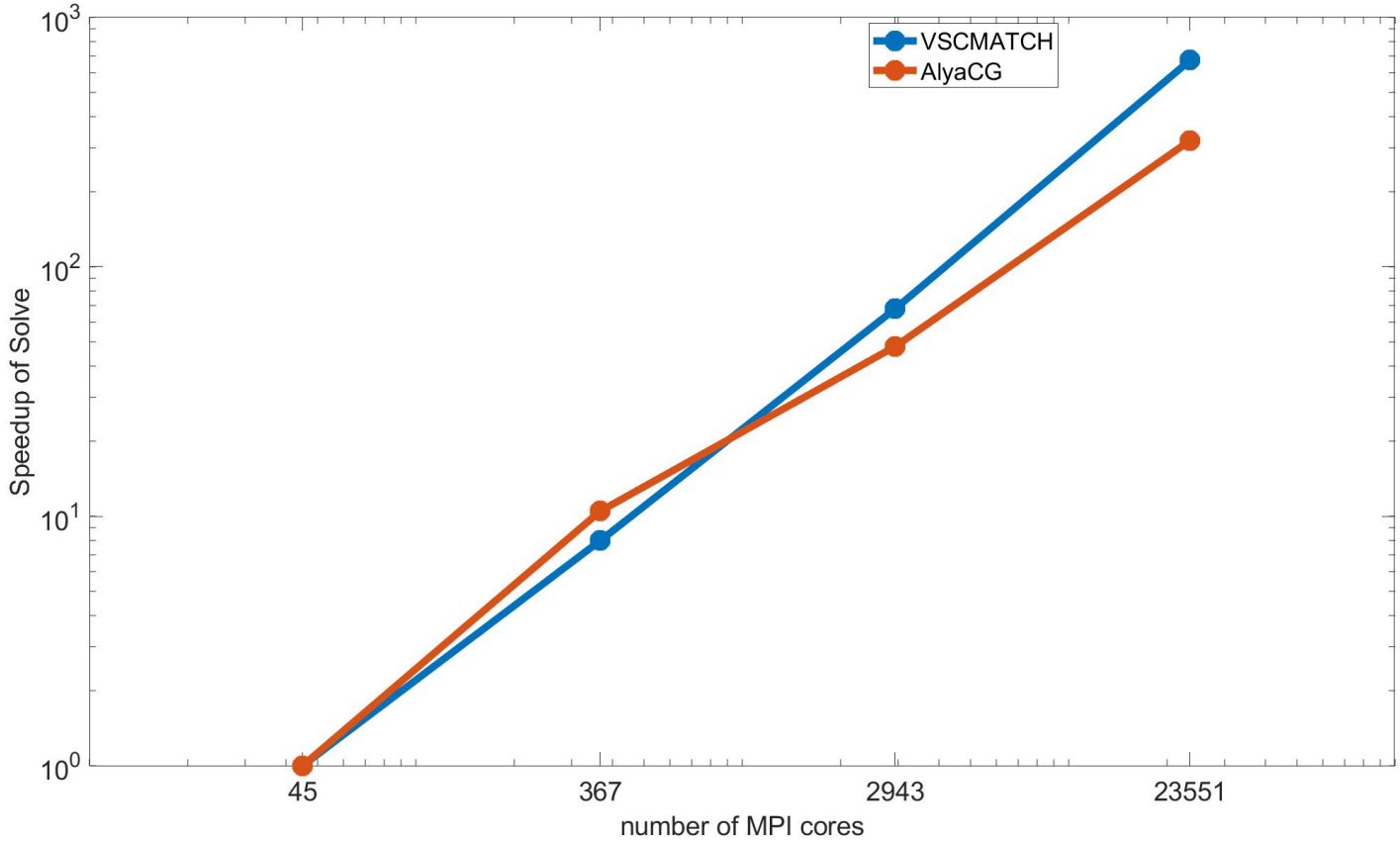
fixed size problem per CPU core $\approx 10^5$ dofs up to 2.9×10^9 dofs
20 time steps in the fully development flow phase



PSCToolkit largely reduces the solve time

Bolund test case - weak scaling on Juwels (JSC) - pressure equation

fixed size problem per CPU core $\approx 10^5$ dofs up to 2.9×10^9 dofs
20 time steps in the fully development flow phase



PSCToolkit improves algorithmic and implementation scalability of Alya

Concluding remarks and work in progress

- PSCToolkit is a software project addressing scalability, flexibility and robusteness for high-performance scientific computing at extreme scale
- our new parallel CMATCH aggregation shows algorithmic and implementation scalability
- we solve systems with size larger than 10^{10} on hybrid pre-exascale computers saving time and energy; comparison with available software demonstrates the validity of our approaches
- integration and testing within very large scale wind simulations and hydrology applications, in collaborations with BSC and JSC, gave very promising results
- we want to explore extreme scalability beyond $10^5/10^6$ computing cores and trillions (10^{12}) of dofs with early access grant to Leonardo also testing on-going work on CA-Krylov solvers and mixed-precision AMG preconditioners

Main references

1. P. D'Ambra, F. Durastante, S. Ferdous, S. Filippone, M. Halappanavar, A. Pothen, AMG Preconditioners based on Parallel Hybrid Coarsening and Multi-objective Graph Matching, Proc. of Euromicro on PDP, 2023.
2. P. D'Ambra, F. Durastante, S. Filippone, Parallel Sparse Computation Toolkit, Software Impacts, Vol. 15, 2023.
3. H. Owen, G. Houzeaux, F. Durastante, S. Filippone, P. D'Ambra, Alya towards Exascale: Algorithmic Scalability using PSCToolkit, 2022, Under revision.
4. P. D'Ambra, F. Durastante, S. Filippone, AMG Preconditioners for Linear Solvers towards Extreme Scale, SIAM Journal on Scientific Computing, Vol. 43, N.5, 2021.
5. M. Bernaschi, P. D'Ambra, D. Pasquini, AMG based on Compatible Weighted Matching on GPUs, Parallel Computing. Vol. 92, 2020.
6. P. D'Ambra, S. Filippone, P. S. Vassilevski, BootCMatch: a Software Package for Bootstrap AMG Based on Graph Weighted Matching, ACM Transactions on Mathematical Software, Vol. 44, 2018.
7. P. D'Ambra, P. S. Vassilevski, Adaptive AMG with coarsening based on compatible weighted matching, Computing and Visualization in Science, Vol. 16, 2013.

Thanks for Your Attention

This work was performed with support of the European Union's Horizon 2020 research and innovation programme under grant agreement N. 824158 and under H2020-JTI-EuroHPC agreement N. 956831