AIgebraic MultiGrid Preconditioners for Sparse Linear Solvers at Extreme Scales on Hybrid Architectures

Salvatore Filippone

Università degli studi di Roma Tor Vergata
salvatore.filippone@uniroma2.it
Consiglio Nazionale delle Ricerche
Istituto per le Applicazioni del Calcolo "M. Picone"

PASC2021
MS: Scalable Solvers for Energy Oriented Scientific Challenges
July 7, 2021

Collaborators and Funding

Joint work with

Pasqua D'Ambra,
Consiglio Nazionale delle Ricerche
Istituto per le Applicazioni del Calcolo "M. Picone"

Fabio Durastante,
Università di Pisa
Consiglio Nazionale delle Ricerche
Istituto per le Applicazioni del Calcolo "M. Picone"

Funded by

E·C·E Project ID: 824158

European Commission | Horizon 2020 European Union funding for Research & Innovation

What we want to solve

$$\text{Solve}: \; Ax = b,$$

where

- $A \in \mathbb{R}^{n \times n}$ is a very large and sparse matrix $\text{nnz}(A) = O(n)$,
- $x, b \in \mathbb{R}^n$,

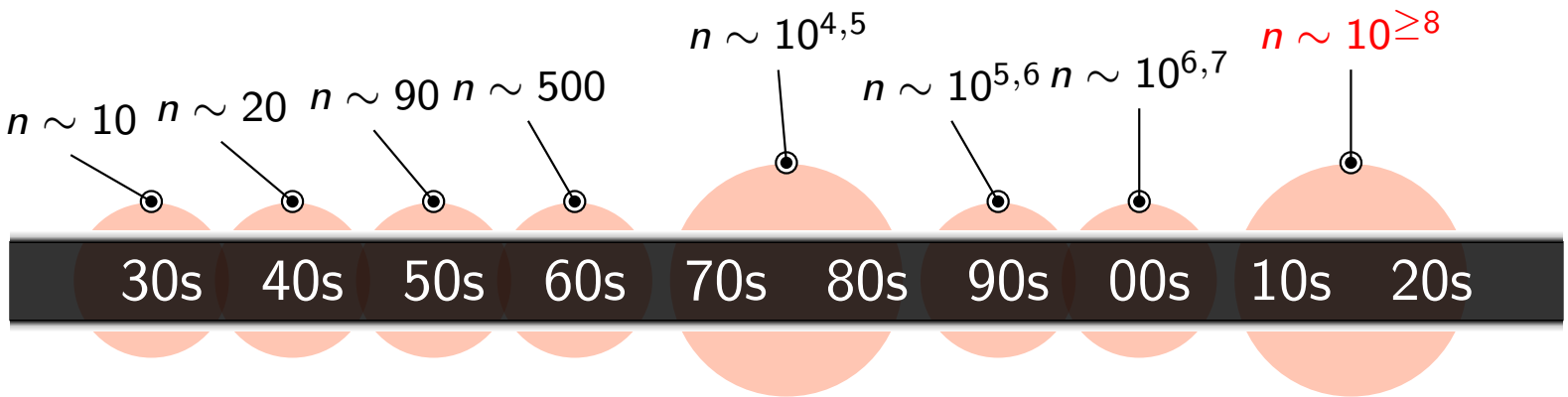
is often the most time consuming computational kernel in many areas of computational science and engineering problems.

What we want to solve

$$\text{Solve}: \; Ax = b,$$

where

- $A \in \mathbb{R}^{n \times n}$ is a very large and sparse matrix $\text{nnz}(A) = O(n)$,
- $x, b \in \mathbb{R}^n$.



The exascale challenge, using computer that perform 10^{15} Flops, targeting next-gen systems performing 10^{18} Flops to solve problems with tens of billions of unknowns.

Target Applications

Wind Models

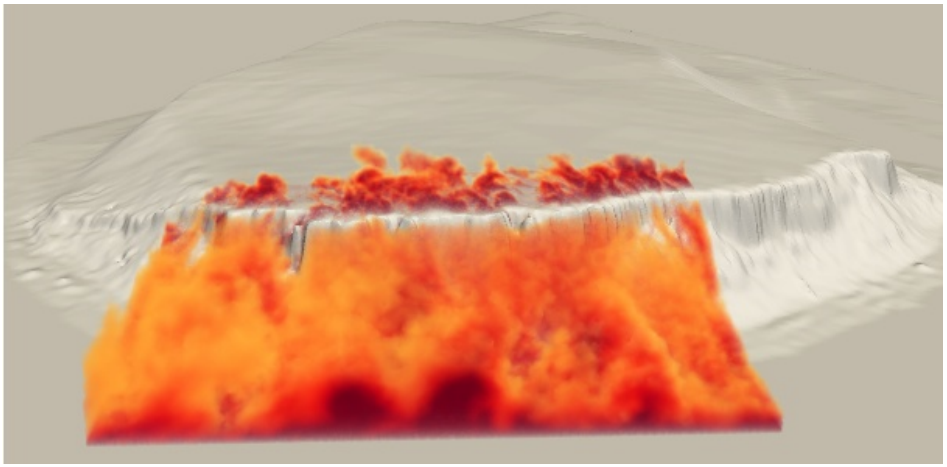
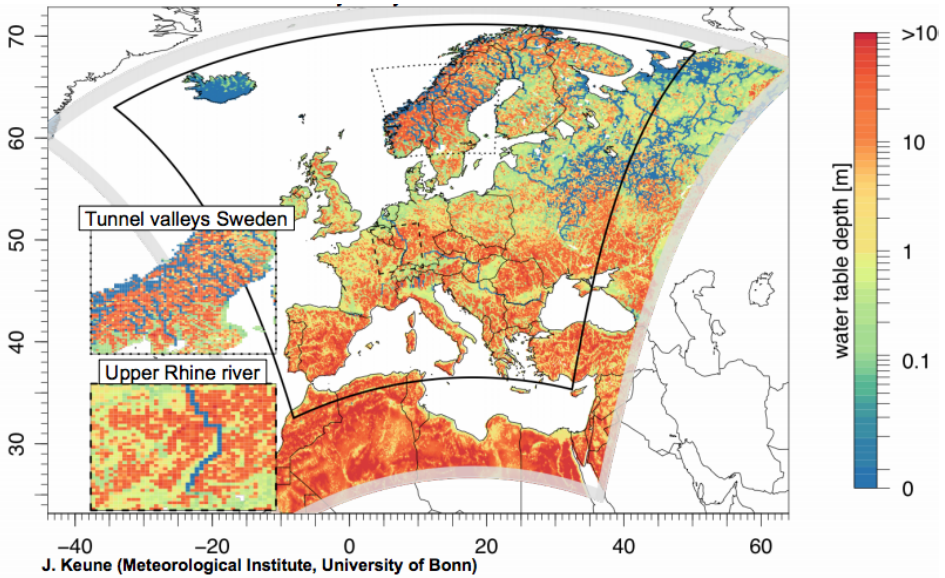


Image credits H. Owen and G. Marin, Barcelona Supercomputing Centre

- Navier-Stokes equations,
- Euler equations,
- Large Eddy Simulations,
- ...

Regional Hydrological Models



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

DoFs: $n \sim 10^{10}$, Processors(cores): $np \sim 10^6$

Where we want to solve it[1]

	System	Cores	Rmax (TFlops/s)
1	Fugaku	7,630,848	442,010.0
2	Summit	2,414,592	148,600.0
3	Sierra	1,572,480	94,640.0
⋮	⋮	⋮	⋮
11	Marconi-100	347,776	21,640.0
12	Piz Daint	387,872	21,230.0
⋮	⋮	⋮	⋮
63	MareNostrum	153,216	6,470.8

MareNostrum IV - BSC

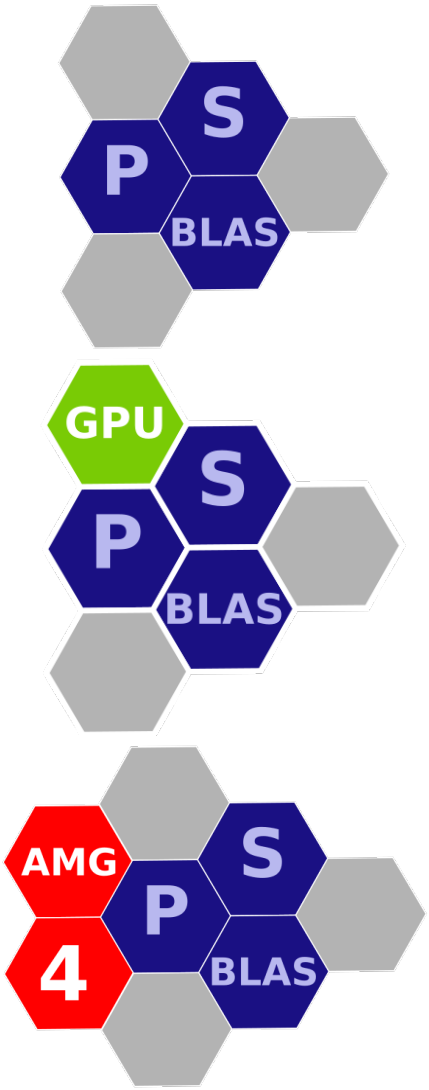
Piz Daint - CSCS

- Machines with thousands of MPI cores,
- Hybrid form of parallelism: MPI, OpenMP, CUDA/OpenCL, ...
- but how we want to solve it?

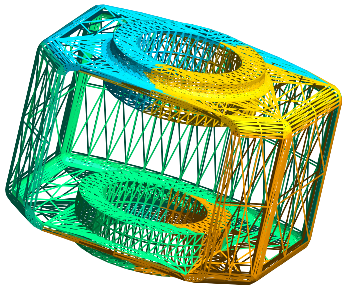
[1] TOP500 list, June 2021 – `https://www.top500.org`

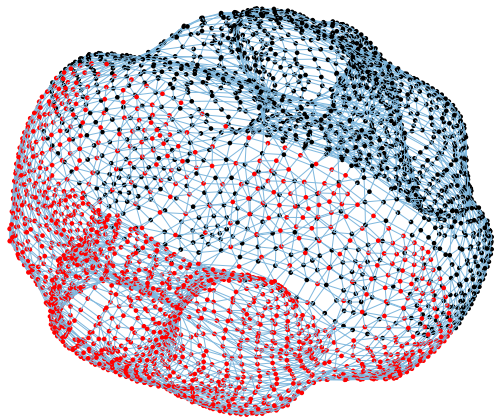
Three central libraries PSBLAS, AMG4PSBLAS and PSBLAS-EXT:

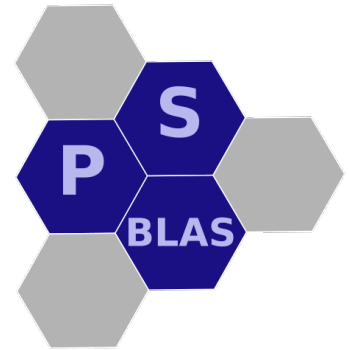
- Existing software standards:
  - MPI, OpenMP, CUDA
  - Serial sparse BLAS,
  - (Par)Metis,
  - AMD
- Attention to performance;
- Research on new preconditioners;
- Data structures are essential, but design for ease of use;
- Tools for large mesh handling: the essential kernel is halo data exchange;
- Krylov subspace solvers;

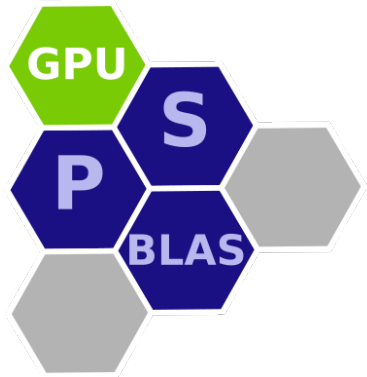


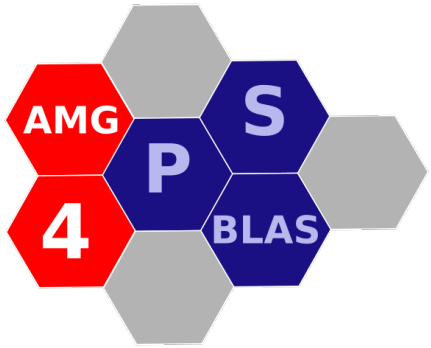
Three central libraries PSBLAS, AMG4PSBLAS and PSBLAS-EXT: Large mesh handling support





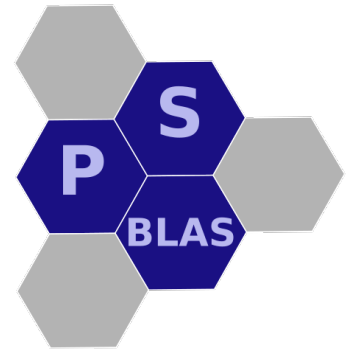


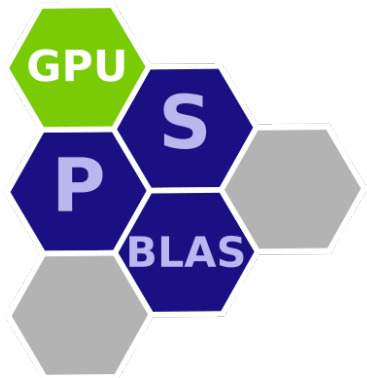


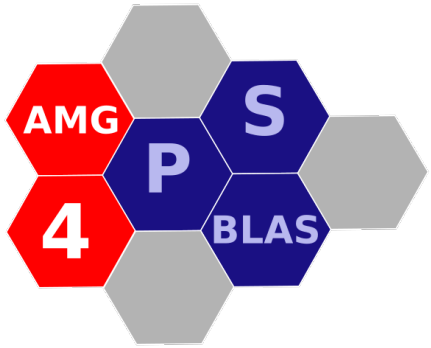


Three central libraries PSBLAS, AMG4PSBLAS and PSBLAS-EXT:

- Domain decomposition preconditioners
- Algebraic multigrid with aggregation schemes
  - Parallel coupled Weighted Matching Based Aggregation
  - Smoothed Aggregation (Vaněk, Mandel, Brezina)
- Parallel Smoothers (Block-Jacobi, Hybrid-GS/SGS/FBGS, ℓ_1 variants) that can be coupled with specialized block (approximate) solvers MUMPS, SuperLU, Incomplete Factorizations (AINV, INVK/L, ILU-type)
- V-Cycle, W-Cycle, K-Cycle

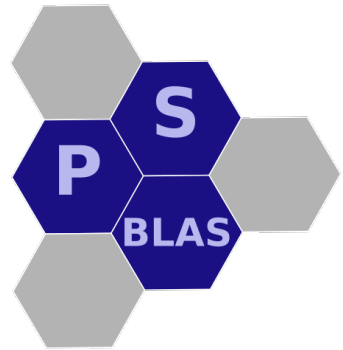


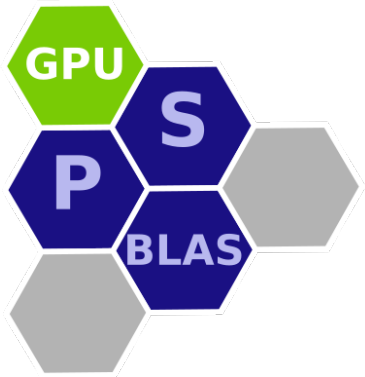


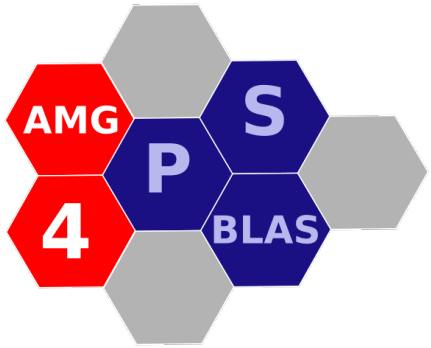


Three central libraries PSBLAS, AMG4PSBLAS and PSBLAS-EXT:

- GPU Plugin PSBLAS-EXT
- Support for NVIDIA devices;
- Many data storage formats;
- Fully integrated in PSBLAS, MPI enabled;
- Transparent use from PSBLAS/AMG4PSBLAS



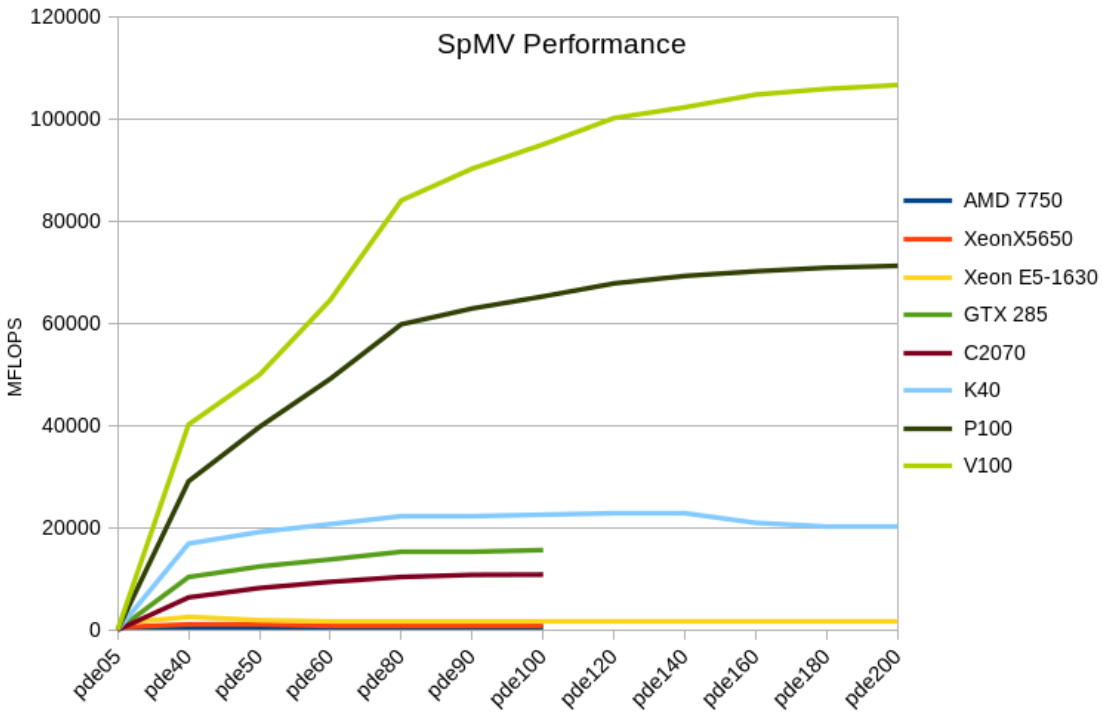


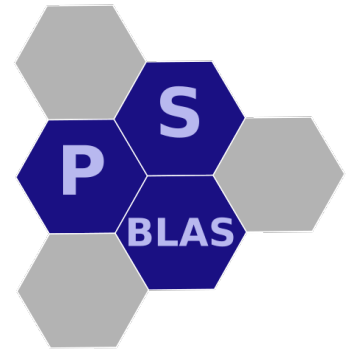


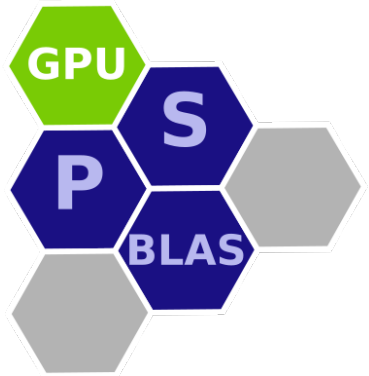
S. Filippone et al., Sparse matrix-vector multiplication on GPGPUs, ACM Trans. Math. Software **43** (2017), no. 4, Art. 30

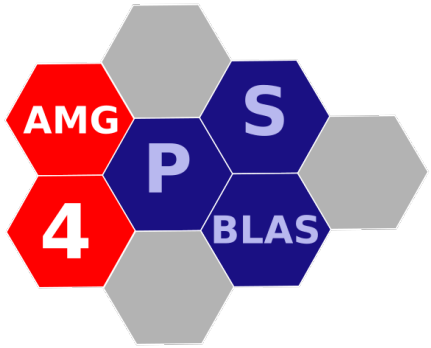
Three central libraries PSBLAS, AMG4PSBLAS and PSBLAS-EXT.

- GPU Plugin PSBLAS-EXT







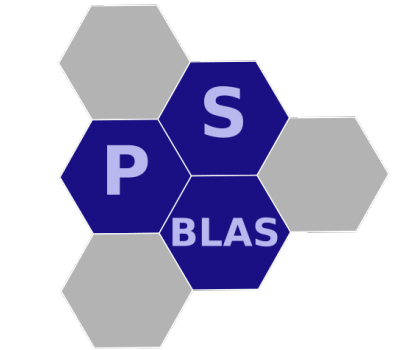


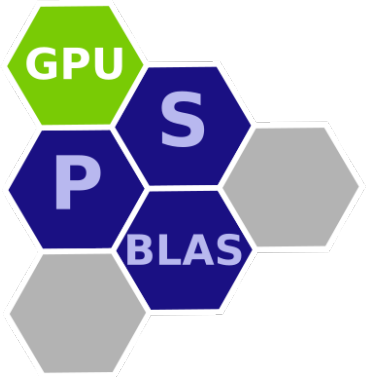
Three central libraries PSBLAS, AMG4PSBLAS and PSBLAS-EXT

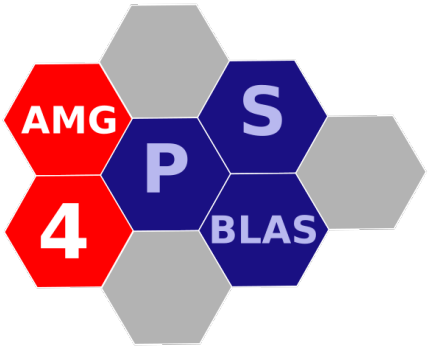
- Freely available from: `https://psctoolkit.github.io`,
- Open Source, released under BSD 3 Clause License,
- Interfaced with the Alya multi-physics solver, ParFlow solver, KINSOL non-linear solvers, collaborations with:

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

JÜLICH Forschungszentrum







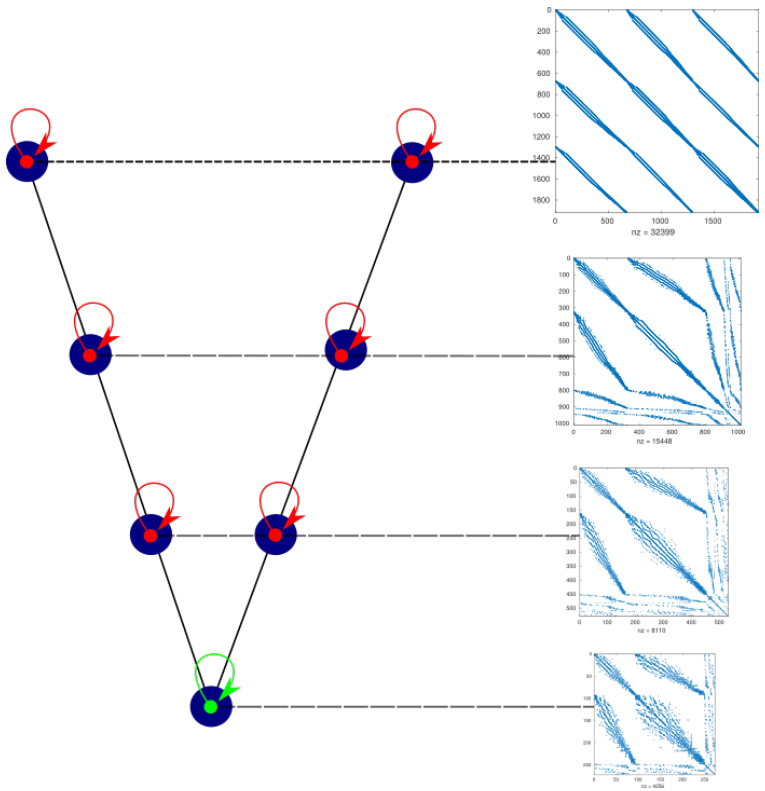
Algebraic Multigrid Preconditioners

Given Matrix $A \in \mathbb{R}^{n \times n}$ SPD

Wanted Iterative method B to precondition the CG method:

- Hierarchy of systems
  $$A_l x_l = b_l, \; l = 0, \ldots, \text{nlev}$$
- Transfer operators:
  $$P_{l+1}^{l} : \mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_l}$$

Missing Structural/geometric infos



Smoother

$$M_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$$

"High frequencies"

Prolongator

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

"Low frequencies"

Complementarity of Smoother and Prolongator

What are we looking for?

Solve the preconditioned system:

$$B^{-1}Ax = B^{-1}b,$$

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

Algorithmic scalability $\max_i \lambda_i(B^{-1}A) \approx 1$ being independent of n,

Linear complexity the action of B^{-1} costs as little as possible, the best being $\mathcal{O}(n)$ flops,

Implementation scalability in a massively parallel computer, B^{-1} should be composed of local actions, performance should depend linearly on the number of processors employed.

What is our *recipe*?

- The smoother M is a standard iterative solver with good parallel properties, e.g., ℓ_1–Jacobi, Hybrid-FBGS, Hybrid-SGS, CG method, etc.
- The prolongator P is built by dofs *aggregation based on matching* in the weighted (adjacency) graph of A.
- The coarse solver can be (again) a preconditioned CG method.

What is our *recipe*?

- The smoother M is an iterative solver with good parallel properties:

  GS $A = M - N$, with $M = L + D$ and $N = -L^T$, where $D = \mathrm{diag}(A)$ and $L = \mathrm{tril}(A)$ is intrinsically sequential!

  HGS Inexact block-Jacobi version of GS, in the portion of the row-block local to each process the method acts as the GS method.

  ℓ_1-HGS On process $p = 1, \ldots, np$ relative to the index set Ω_p we factorize $A_{pp} = L_{pp} + D_{pp} + L_{pp}^T$ for $D_{pp} = \mathrm{diag}(A_{pp})$ and $L_{pp} = \mathrm{tril}(A_{pp})$ then:

$$
\begin{aligned}
M_{\ell_1-HGS} &= \mathrm{diag}((M_{\ell_1-HGS})_p)_{p=1,\ldots np}, \\
(M_{\ell_1-HGS})_p &= L_{pp} + D_{pp} + D_{\ell_1 p}, \\
(d_{\ell_1})_{i=1}^{nb} &= \sum_{j \in \Omega_p^{nb}} |a_{ij}|.
\end{aligned}
$$

$$M_{\ell_1-HGS} = \mathrm{diag}((M_{\ell_1-HGS})_p)_{p=1,\ldots np},$$

  AINV Block-Jacobi with an approximate inverse factorization on the block $\Rightarrow$ suitable for GPUs

What is our *recipe*?

- The prolongator P is built by dofs *aggregation based on matching* in the weighted (adjacency) graph of A.

Given w $\in \mathbb{R}^n$, let $P \in \mathbb{R}^{n \times n_c}$ and $P_f \in \mathbb{R}^{n \times n_f}$ be a prolongator and a complementary prolongator, such that:

$$\mathbb{R}^n = \text{Range}(P) \oplus^{\perp} \text{Range}(P_f), \quad n = n_c + n_f$$

w $\in$ Range(P): coarse space Range(P_f): complementary space

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

A_c: coarse matrix A_f: hierarchical complement

Sufficient condition for efficient coarsening

$A_f = P_f^T A P_f$ as well conditioned as possible, i.e.,
Convergence rate of *compatible relaxation*: $\rho_f = \|I - M_f^{-1} A_f\|_{A_f} \ll 1$

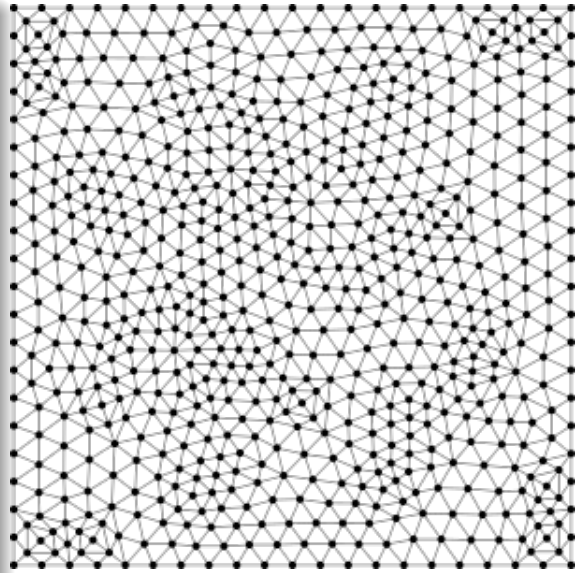
But *how* we achieve it?

Weighted graph matching

Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight vector 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 if it maximizes the product of the weights of the edges $e_{i \mapsto j}$ in it.



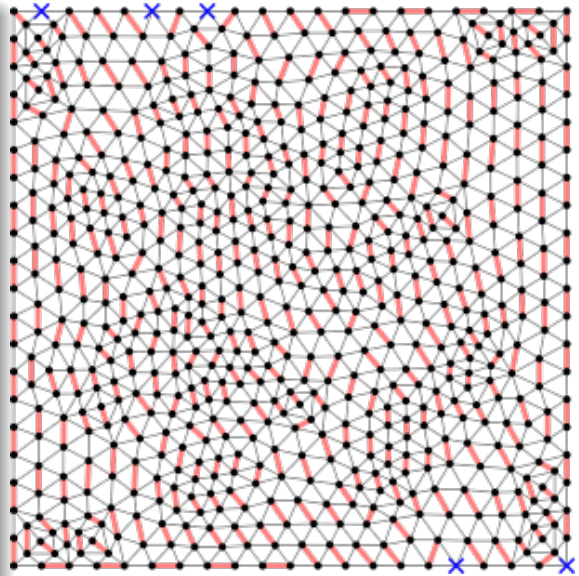
But *how* we achieve it?

Weighted graph matching

Given a graph $G = (\mathcal{V}, \mathcal{E})$ (with adjacency matrix A), and a weight vector 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 if it maximizes the product of the weights of the edges $e_{i \mapsto j}$ in it.

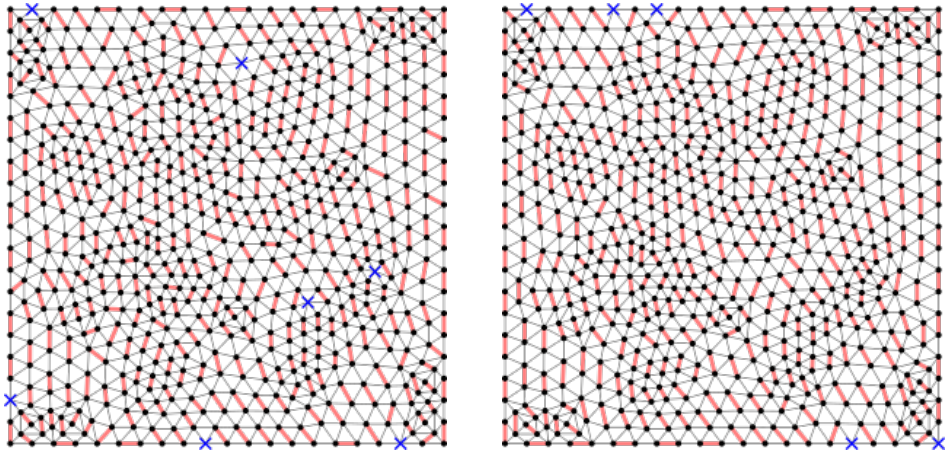


We divide the index set into matched vertices $\mathcal{I} = \bigcup_{i=1}^{n_p} \mathcal{G}_i$, with $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ if $i \neq j$, and unmatched vertices, i.e., n_s singletons G_i.

Parallel Matching Algorithms

1. What is the best matching algorithm from a computational point of view?
2. How can we evaluate the quality (in term of the AMG algorithm) of the resulting matching?

With the formalism from (Xu and Zikatanov, 2017) and using a technique from (Napov and Notay, 2011) we associate a quality measure of the aggregates in terms of the convergence properties of the whole AMG method! Better aggregates give better convergence properties.



Algorithm: Locally Dominant Edge

Input: Graph $G = (\mathcal{V}, \mathcal{E})$, Weights $\hat{A}$

$\mathcal{M} \leftarrow \emptyset$;

while $\mathcal{E} \neq \emptyset$ **do**

  Take a locally dominant edge $(i, j) \in \mathcal{E}$, i.e., such that

$$\arg\max_k \hat{a}_{ik} = \arg\max_k \hat{a}_{jk} = \hat{a}_{ij}$$

  Add $(i, j) \in \mathcal{M}$;

  Remove all edges incident to i and j from $\mathcal{E}$;

end

Output: Matching $\mathcal{M}$

Weak Scalability - CPU/GPU Runs - Piz Daint

- Run on the Piz Daint machine up to 28800 cores
- Test: 3D Constant coefficient Poisson Problem with FCG
- DoF: 256k/512k/1M unknowns $\times$ MPI core
- Measure: Solve Time (s).

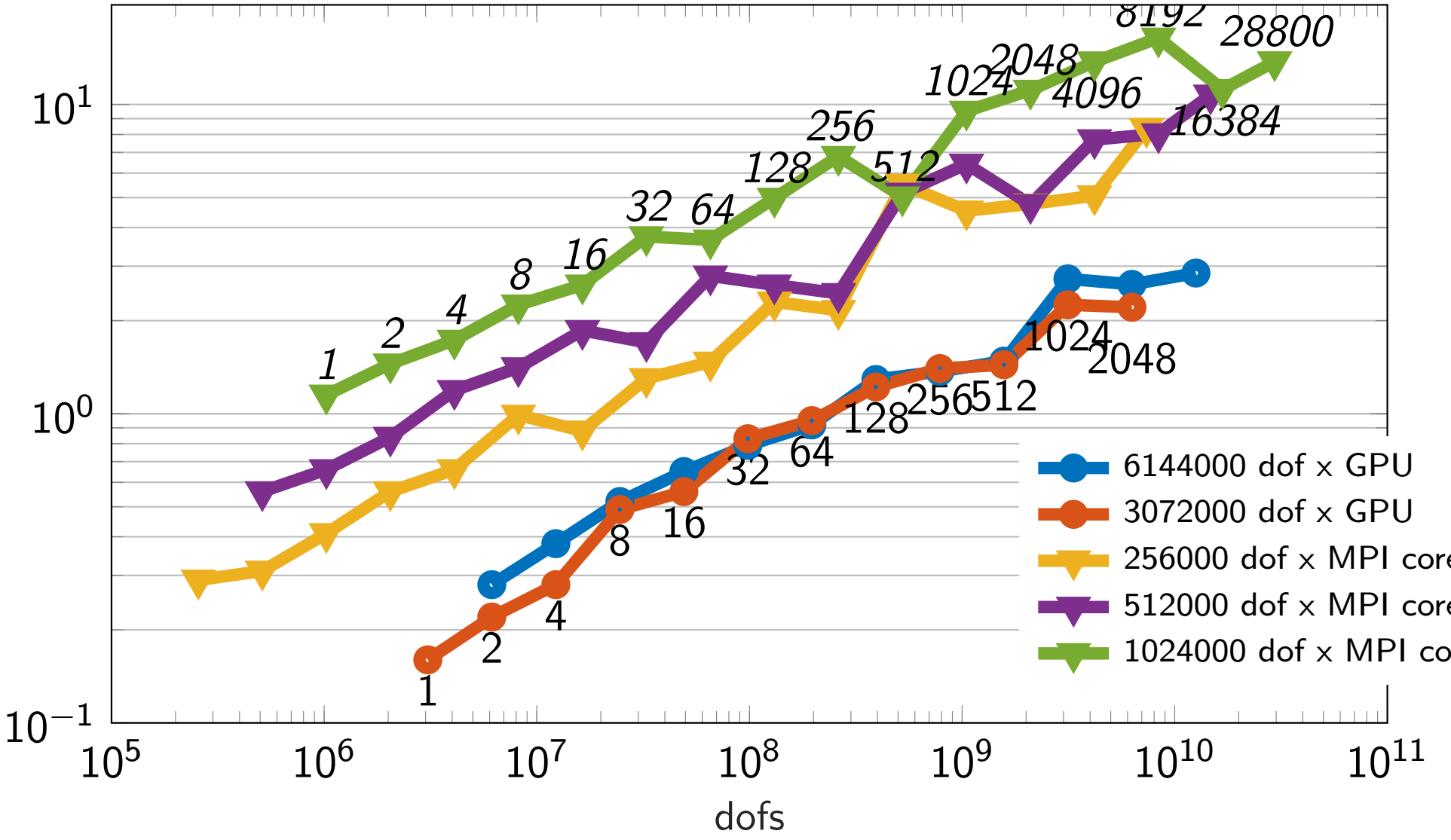
Scaling

There are two common notions of scalability:

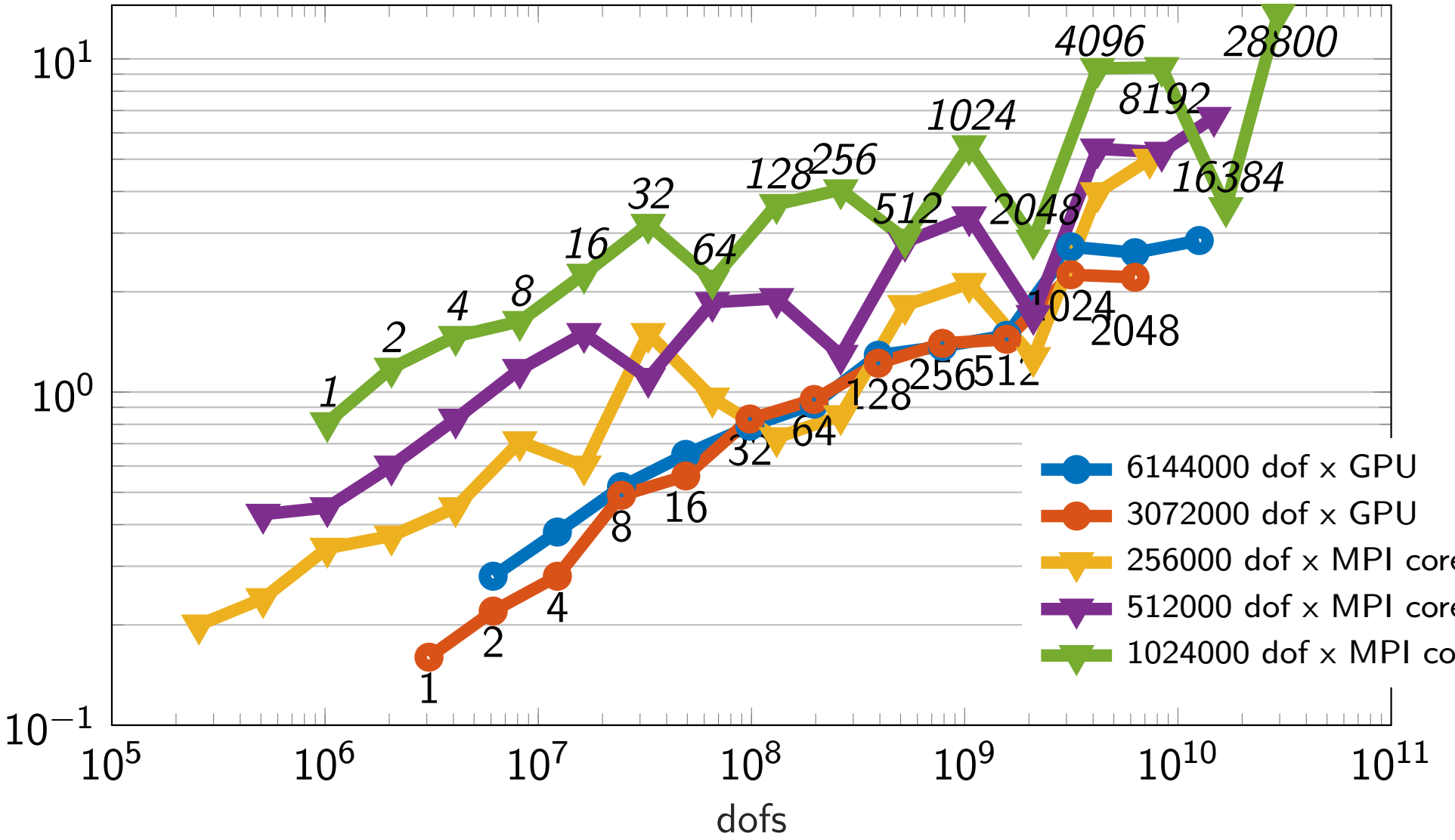
- Strong scaling analysis studies as how the solution time varies with the number of processors for a fixed total problem size.
- Weak scaling analysis studies as how the solution time varies with the number of processors for a fixed problem size per processor.

P. D'Ambra, F. Durastante and S. Filippone,AMG preconditioners for Linear Solvers towards Extreme Scale. arXiv preprint (2020), arXiv:2006.16147, to appear in SIAM J. Sci. Comp. 2021

Weak Scalability - CPU/GPU Runs - Piz Daint

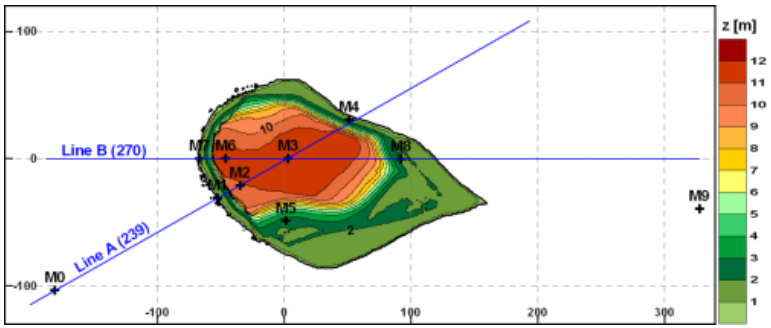


Weak Scalability - CPU/GPU Runs - Piz Daint



A CFD application inside Alya

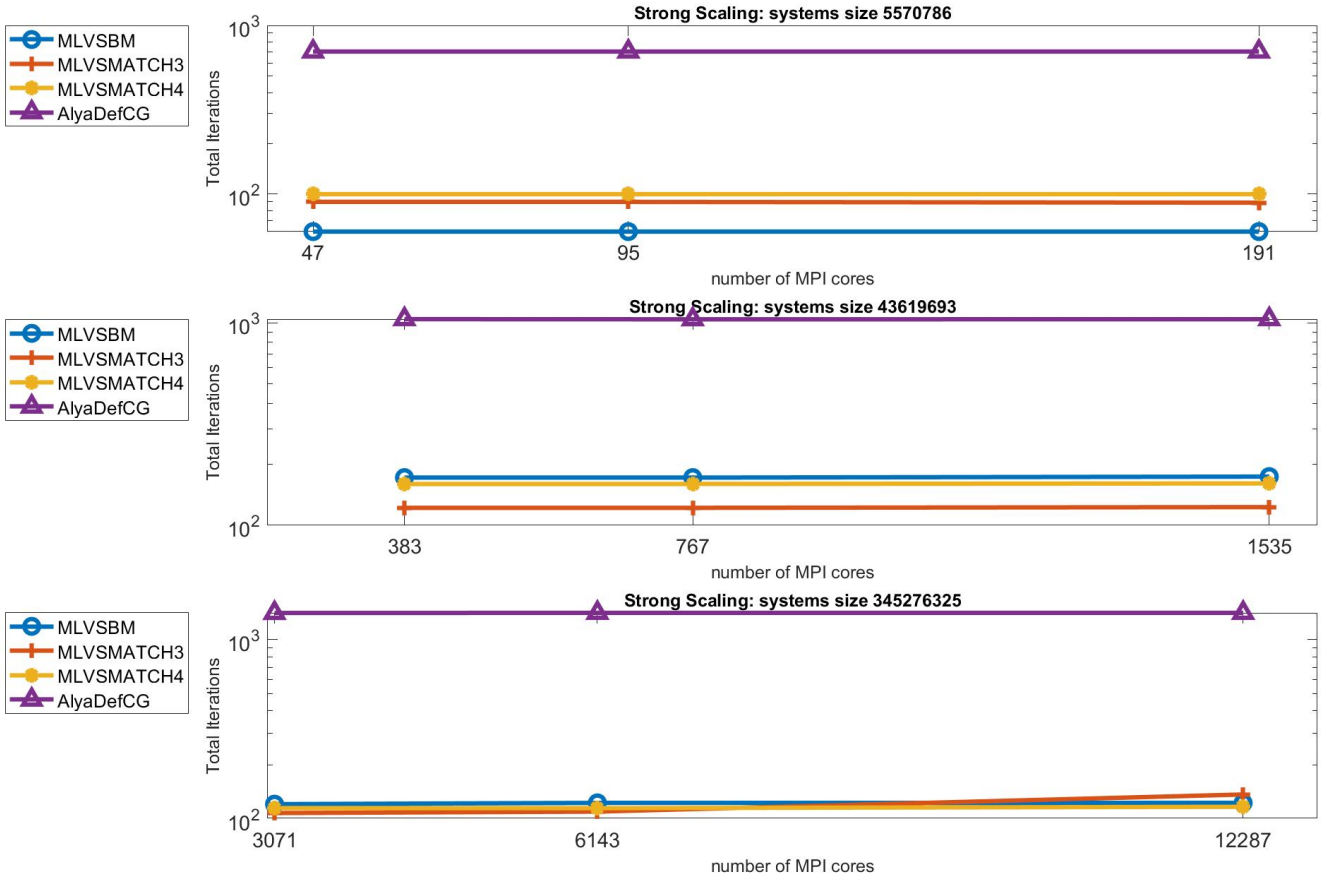
Joint work with
Herbert Owen
Barcelona Super Computing Center



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.

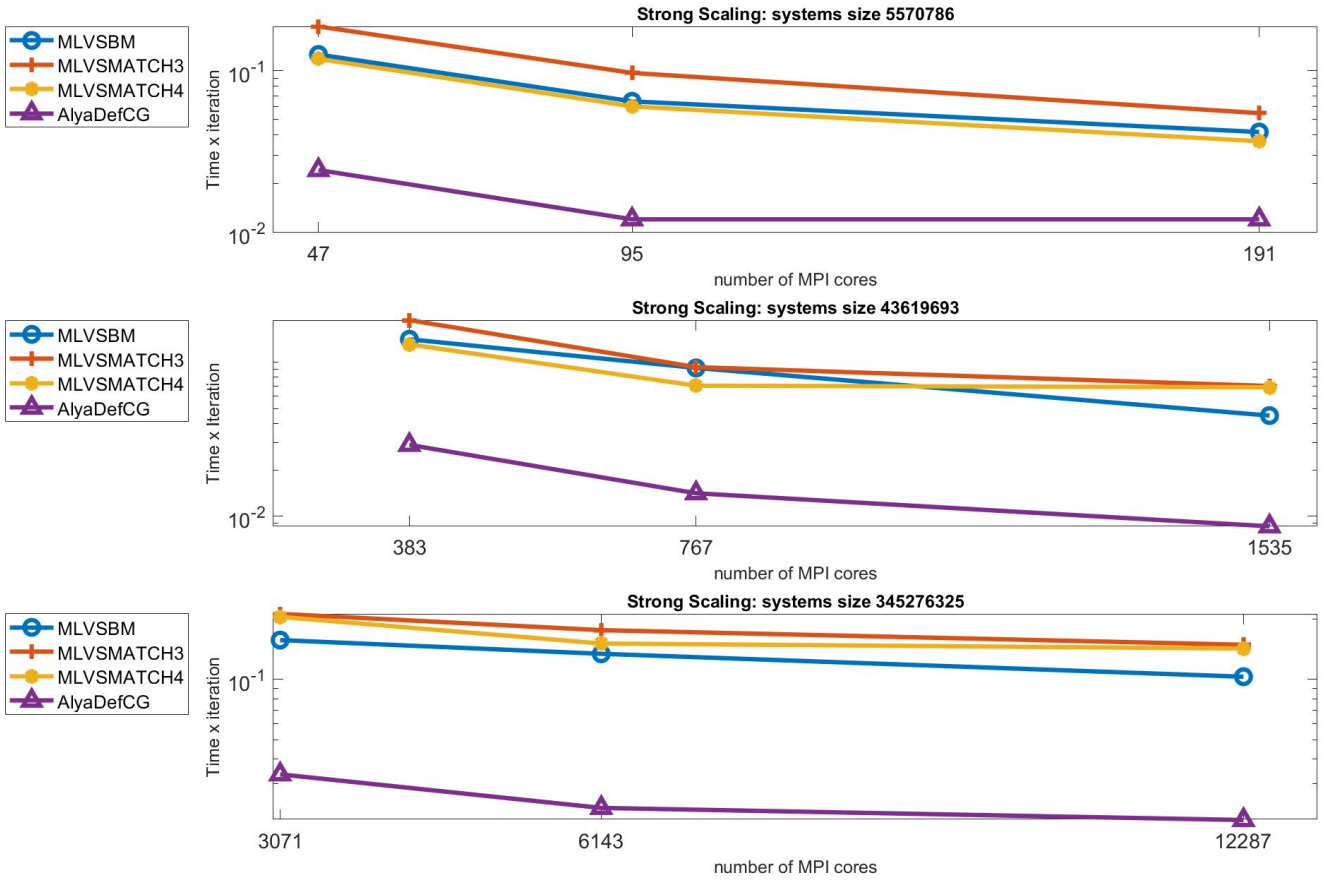
- **Model**: 3D incompressible unsteady Navier-Stokes equations for the Large Eddy Simulations of turbulent flows,
- **Discretization**: low-dissipation mixed FEM (linear FEM both for velocity and pressure),
- **Time-Stepping**: non-incremental fractional-step for pressure, explicit fourth order Runge-Kutta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

Bolund Test Case - Strong Scaling - Pressure Equation



- Total number of linear iterations is smaller and stable for increasing number of cores,

Bolund Test Case - Strong Scaling - Pressure Equation



- The time needed per each iteration decreases for increasing number of cores,
- The trade-off between cost-per-iteration and number of iterations advantages the AMG preconditioners!

Conclusions and Future Directions

We have proved

- ✔ Aggregation procedure with certified quality,
- ✔ Scalability results on tens of thousands of cores,
- ✔ Comparable results with state of the art libraries,
- ✔ Interfacing with large scale scientific applications,
- ✔ Multi-GPU support.

Algorithmic and software extensions to AMG4PSBLAS (future work)

- ⚙ Multi-objective matching to increase coarsening ratio,
  - 👥 Collaboration with Pacific Northwest National Laboratory (Richland, WA), and Purdue University (IN)
- ⚙ Process remapping for coarse grid solutions,
  - 👥 Collaboration with Centre national de la recherche scientifique (Toulouse)
- ⚙ GPU data and preconditioner setup improvements,
- ⚙ Communication avoiding Krylov methods,
- ⚙ Mixed-precision arithmetic.

Essential bibliography

- Multigrid based on matching
  - P. D'Ambra and P. S. Vassilevski, Adaptive AMG with coarsening based on compatible weighted matching, Comput. Vis. Sci. **16** (2013), no. 2, 59–76.
  - P. D'Ambra, S. Filippone and P. S. Vassilevski, BootCMatch: a software package for bootstrap AMG based on graph weighted matching, ACM Trans. Math. Software **44** (2018), no. 4, Art. 39, 25 pp.
  - M. Bernaschi, P. D'Ambra and D. Pasquini, AMG based on compatible weighted matching for GPUs, Parallel Comput. **92** (2020), 102599, 13 pp.
  - P. D'Ambra, F. Durastante and S. Filippone, On the quality of matching-based aggregates for algebraic coarsening of SPD matrices in AMG. arXiv preprint (2020), `arXiv:2001.09969`.
- Scalability results
  - P. D'Ambra, F. Durastante and S. Filippone,AMG preconditioners for Linear Solvers towards Extreme Scale. To appear in Siam J. Sci. Comput (2021). arXiv preprint, `arXiv:2006.16147`.
- PSBLAS
  - S. Filippone and A. Buttari, Object-oriented techniques for sparse matrix computations in Fortran 2003. ACM Trans. Math. Software **38** (2012), no. 4, 1–20 pp.
  - S. Filippone et al., Sparse matrix-vector multiplication on GPGPUs, ACM Trans. Math. Software **43** (2017), no. 4, Art. 30, 49 pp.

Thank You!

Quality and Convergence: *a posteriori* analysis

Convergence Theorem (D'Ambra, Durastante, Filippone)

The exact TL–AMG with convergent smoother M, and prolongator P based on the maximum weight matching applied on a SPD matrix A has a convergence rate of

$$\|I - B^{-1}A\|_A \leq 1 - \frac{\mu_c}{c^D}, \text{ for } \mu_c = \min_{1\leq j\leq J} \mu_j(V_j^c) = \min_{1\leq j\leq J} \left[\max_{\mathrm{v}_j \in V_j} \min_{\mathrm{v}_j^c \in V_j^c} \frac{\|\mathrm{v}_j - \mathrm{v}_j^c\|_{D_j}^2}{\|\mathrm{v}_j\|_{A_j}^2} \right].$$

and c^D the continuity constant of the smoother. Moreover, the $\mu_j^{-1}(V_j^c)$ are such that

$$\lambda_2^{-1}(D_j^{-1}A_j) \leq \mu_j^{-1}(V_j^c) \leq \lambda_1^{-1}(D_j^{-1}A_j).$$

Furthermore, if either $(\mathrm{w}_{e_{i\to j}}, \lambda_1(D_j^{-1}A_j)$, or $(\mathrm{w}_{e_{i\to j}}^{\perp}, \lambda_2(D_j^{-1}A_j)$ are eigencouples of $D_j^{-1}A_j$, then

$$\mu_j^{-1}(V_j^c) = \lambda_2^{-1}(D_j^{-1}A_j)$$

- The local constants $\mu_j^{-1}(V_j^c)$ are then a quality measure for the single aggregates

Fixing the parameters

We can fix the weight vector w, and evaluate the performance of the matching algorithms.

Theorem (Optimal prolongator)

Let $\{\lambda_j, \Phi_j\}_{j=1}^{n}$ be the eigenpairs of $\overline{T} = \overline{M}A$ for the symmetrized smoother $\overline{M}$. Let us also assume that Φ_j are orthogonal w.r.t. $(\cdot,\cdot)_{\overline{M}^{-1}}$. The convergence rate $\|E(P)\|_A$ is minimal for P such that

$$\text{Range}(P) = \text{Range}(P^{opt}),$$

where $P^{opt} = \{\Phi_1, \ldots, \Phi_{n_c}\}$. In this case,

$$\|E\|_A^2 = 1 - \lambda_{n_c+1}$$

A good candidate can be obtained by exploiting the symmetrized smoother $\overline{M}$ to select as a weight vector an ε–smooth algebraic vector, i.e., for a given $\epsilon \in (0,1)$, v an algebraically ϵ-smooth with respect to A if

$$\|\mathrm{v}\|_A^2 \leq \epsilon \|\mathrm{v}\|_{\overline{M}^{-1}}^2.$$

For our choice of P we know that:

- There exists $\mathrm{h} \in \mathbb{R}^{n_c}$ such that $P\mathrm{h} = \mathrm{w}$