# AMG Preconditioners for Computational and Data Science at Extreme Scale

Pasqua D'Ambra

Institute for Applied Computing of the National Research Council (IAC-CNR), Naples, IT
pasqua.dambra@cnr.it

Calcolo Scientifico e Modelli Matematici

April 6-8, 2022

# The HPC Team at

Projects Participants :

- Massimo Bernaschi (IAC-CNR), IT
- Salvatore Filippone (Univ. of Rome Tor-Vergata and IAC-CNR), IT
- Fabio Durastante (Univ. of Pisa and IAC-CNR), IT
- Mauro Carrozzo (IAC-CNR), IT
- Alessandro Celestini (IAC-CNR), IT
- Gabriele Salvati (IAC-CNR), IT
- Daniele Bertaccini (Univ. of Rome Tor-Vergata and IAC-CNR), IT

Collaborations :

Mahantesh M. Halappanavar, PNNL (Richland, WA), USA

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

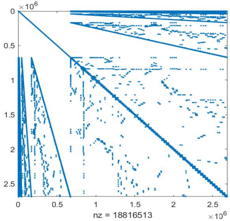
Ludmil Zikatanov, The Penn State University, PSU (State College), USA

# What we want to solve

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

$n$ **large**

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



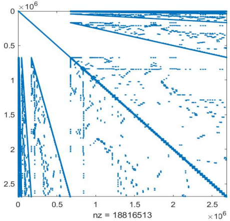
often the most time consuming computational kernel in many areas of Computational/Data Science

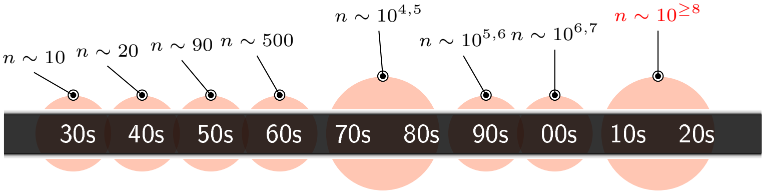
# What we want to solve

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

$n$ **large**

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





$n \sim 10$

$n \sim 20$

$n \sim 90$

$n \sim 500$

$n \sim 10^{4,5}$

$n \sim 10^{5,6}$

$n \sim 10^{6,7}$

$n \sim 10^{\geq 8}$

30s 40s 50s 60s 70s 80s 90s 00s 10s 20s

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

# 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 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 18 | Marconi-100 | 347,776 | 21,640.0 |
| 20 | Piz Daint | 387,872 | 21,230.0 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 74 | MareNostrum | 153,216 | 6,470.8 |

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

Marconi 100 - Cineca

Piz Daint - CSCS

[1] TOP500 list, November 2021 – `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
- 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
- minimizing energy consumption is important for sustainability of HPC

New Mathematics, new algorithms

and new software development tools are needed

# 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
- minimizing energy consumption is important for sustainability of HPC

LA FRONTIERA

Il software matematico: le sfide della ricerca sui supercalcolatori

Home > Cultura E Società Digitali

Il software matematico è alla base della simulazione computazionale applicata nei più diversi settori, dalla fluidodinamica all'Intelligenza Artificiale, ed è determinante per lo sviluppo dei supercalcolatori. Cos'è, come funziona, storia, strumenti e metodologie

25 Feb 2022

**Pasqua D'Ambra**
CNR, IAC, CINI HPC-KTT

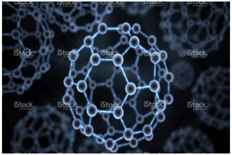
**Salvatore Filippone**
università di Roma "Tor-Vergata" , CINI HPC-KTT

# 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

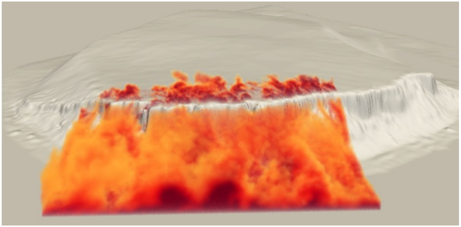
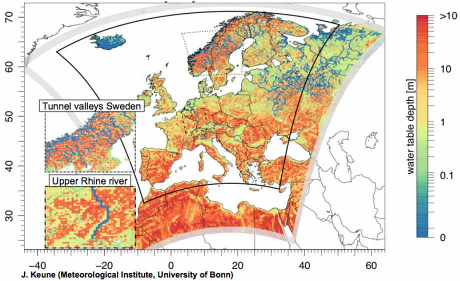


Image credits H. Owen and G. Marin, Barcelona Supercomputing Centre

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

## Regional Hydrological Models



J. Keune (Meteorological Institute, University of Bonn)

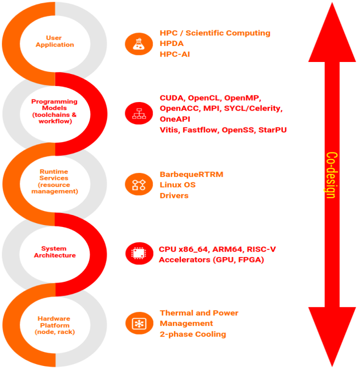
- Darcy equation,
- Richards equation,
- Equations for overland flow
- ...

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

# TEXTAROSSA project

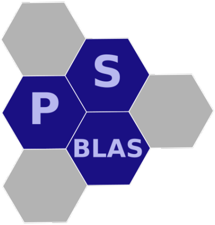
## Towards EXtreme scale Technologies and Accelerators for euROhpc hw/Sw Supercomputing Applications for exascale

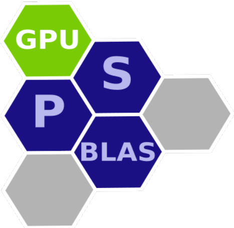
developing new software tools for high-performance and high-energy efficiency on near-future exascale computing systems by multi-directional co-design approach


User Application
HPC / Scientific Computing
HPDA
HPC-AI
Programming Models (toolchains & workflow)
CUDA, OpenCL, OpenMP,
OpenACC, MPI, SYCL/Celerity,
OneAPI
Vitis, Fastflow, OpenSS, StarPU
Runtime Services (resource management)
BarbequeRTRM
Linux OS
Drivers
System Architecture
CPU x86_64, ARM64, RISC-V
Accelerators (GPU, FPGA)
Hardware Platform (node, rack)
Thermal and Power
Management
2-phase Cooling
Co-design

Our contribution: performance/power efficient MathLib

# Parallel Sparse Computation Toolkit



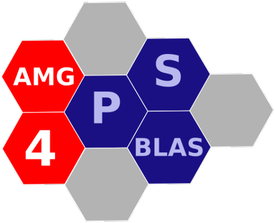


**Parallel Sparse Basic Linear Algebra Subroutines**

Prompted by some works of Iain Duff et al. on standard for Sparse BLAS

+

**its GPU-plugin**



**AMG Preconditioners for PSBLAS**

Available from https://psctoolkit.github.io/

Recently selected as Key Innovation from EU Innovation Radar

# AMG4PSBLAS: AMG Preconditioners for PSBLAS

A software development project started in late 2007

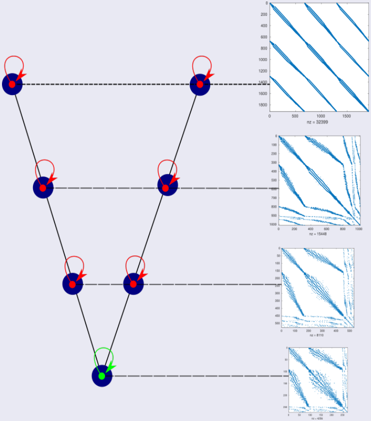
- initially developed as a package of algebraic multigrid Schwarz preconditioners, extended to more general AMG preconditioners
- object-oriented design in Fortran 2003/2008, layered sw architecture on top of PSBLAS
  $\Longrightarrow$ modularity and flexibility
- clear separation between interface and implementation of methods
  $\Longrightarrow$ performance and extensibility (e.g., works transparently on GPUs)
- separated users' interface for setup of the multigrid hierarchy and setup of the smoothers and solvers to have large flexibility at each level

P. D'Ambra et al., MLD2P4: a Package of Parallel Algebraic Multilevel Domain Decomposition Preconditioners in Fortran 95, ACM TOMS, 37, 3, 2010

# AMG Methods

## Example: symmetric V-cycle

```
procedure V-cycle$(k, nlev, A^k, b^k, x^k)$
    if $(k \neq nlev)$ then
        $x^k = x^k + (M^k)^{-1}(b^k - A^k x^k)$
        $b^{k+1} = (P^{k+1})^T\,(b^k - A^k x^k)$
        $x^{k+1} =$ V-cycle$(k+1, A^{k+1}, b^{k+1}, 0)$
        $x^k = x^k + P^{k+1} x^{k+1}$
        $x^k = x^k + (M^k)^{-T}(b^k - A^k x^k)$
    else
        $x^k = \left(A^k\right)^{-1} b^k$
    endif
    return $x^k$
end
```



AMG methods do not explicitly use the problem geometry and rely only on matrix entries to generate coarse grids (setup phase)

# (Two-grid) Convergence

## Theorem

*(Falgout and Vassilevski, 2004) The two-grid preconditioner* $B_{TG}$ *defined from the iteration matrix:*

$$I - B_{TG}^{-1}A = (I - M^{-1}A)(I - P(P^TAP)^{-1}P^TA)(I - M^{-T}A)$$

*is spectrally equivalent to* $A$ *and the following estimate holds:*

$$v^TAv \leq v^TB_{TG}v \leq K_{TG}v^TAv, \text{ with}$$

$$K_{TG} = sup_{v \in \mathcal{R}^n \setminus 0} \frac{v^T\tilde{M}(I - \pi_{\tilde{M}})v}{v^TAv},$$

*where* $\tilde{M} = M(M + M^T - A)^{-1}M^T$ *is the symmetrized smoother and* $\pi_{\tilde{M}} = P(P^T\tilde{M}P)^{-1}P^T\tilde{M}$ *is the* $\tilde{M}-$*based projection.*

# (Two-grid) Convergence

## Theorem

*(Falgout and Vassilevski, 2004) The two-grid preconditioner $B_{TG}$ defined from the iteration matrix:*

$$I - B_{TG}^{-1}A = (I - M^{-1}A)(I - P(P^TAP)^{-1}P^TA)(I - M^{-T}A)$$

*is spectrally equivalent to $A$ and the following estimate holds:*

$$v^TAv \le v^TB_{TG}v \le K_{TG}v^TAv, \text{ with}$$

$$K_{TG} = sup_{v \in \mathcal{R}^n \setminus 0} \frac{v^T\tilde{M}(I - \pi_{\tilde{M}})v}{v^TAv},$$

*where $\tilde{M} = M(M + M^T - A)^{-1}M^T$ is the symmetrized smoother and $\pi_{\tilde{M}} = P(P^T\tilde{M}P)^{-1}P^T\tilde{M}$ is the $\tilde{M}-$based projection.*

## Ideal prolongator

Assume that $R$ and $S$ form an orthogonal decomposition of $\mathcal{R}^n$, i.e., $RS = 0$, and $P$ is such that $PR$ is a projection onto $\mathcal{R}ange(P)$, the best constant is $K^* = (\lambda_{\text{min}}((S^T\tilde{M}S)^{-1}S^TAS))^{-1}$ and the corresponding minimizer is $P^* = (I - S(S^TAS)^{-1}S^TA)R^T$

# Scalable preconditioners

Solve the system:

$$BAx = Bb,$$

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

- $\max_i \lambda_i(B^{-1}A) \approx 1$ being independent of $n$ (algorithmic scalability)
- the action of $B$ costs as little as possible, the best being $\mathcal{O}(n)$ flops (linear complexity)
- in a massively parallel computer, $B$ should be composed of local actions, **(implementation scalability, i.e., performance linearly proportional to the number of processors employed)**

# Scalable preconditioners

Solve the system:

$$BAx = Bb,$$

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

- $\max_i \lambda_i(B^{-1}A) \approx 1$ being independent of $n$ (algorithmic scalability)
- the action of $B$ costs as little as possible, the best being $\mathcal{O}(n)$ flops (linear complexity)
- in a massively parallel computer, $B$ should be composed of local actions, **(implementation scalability, i.e., performance linearly proportional to the number of processors employed)**

## MultiGrid performance parameters

- convergence rate $\rho < 1$: affects number of solver iterations
- operator complexity $opc = \frac{\sum_{k=0}^{nlev-1} nnz(A^k)}{nnz(A^0)}$: affects memory requirements and cycle time
- average stencil size $s(A^k) = nnz\_row(A^k)$: affects computation and communication both in setup and in cycle time

# Our recipies: CMATCH parallel coarsening

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 $\qquad$ $\mathcal{R}ange(P_f)$ : 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 $\qquad$ $A_f$ : hierarchical complement

# Our recipies: CMATCH parallel coarsening

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 $\qquad \mathcal{R}ange(P_f)$ : 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 $\qquad A_f$ : hierarchical complement

## Efficient coarsening (Falgout and Vassilevski, 2004)

Good convergence rate of compatible relaxation:
$\rho_f = \|I - M_f^{-1} A_f\|_{A_f} << 1$ with $M_f = P_f^T M P_f$

# Our recipies: CMATCH parallel coarsening

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

## Efficient coarsening (Falgout and Vassilevski, 2004)

Good convergence rate of compatible relaxation:
$\rho_f = \|I - M_f^{-1} A_f\|_{A_f} << 1$ with $M_f = P_f^T M P_f$

## Our idea (D'Ambra et al., 2013, 2016, 2018)

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**

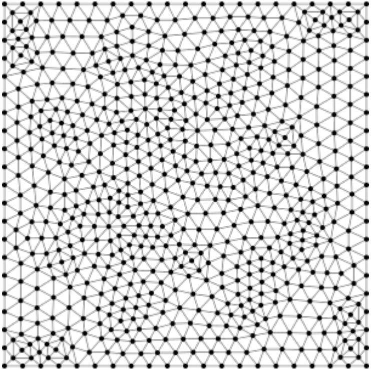
# CMATCH (cont'd)

## Weighted graph matching

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

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

- a *matching* $\mathcal{M}$ is a set of pairwise non-adjacent edges
- a maximum weight matching maximizes the sum of the weights of its edges $e_{i \mapsto j}$



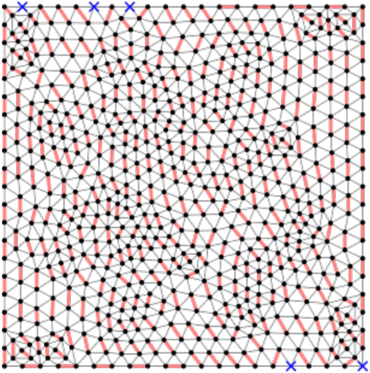
# CMATCH (cont'd)

## Weighted graph matching

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

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

- a *matching* $\mathcal{M}$ is a set of pairwise non-adjacent edges
- a maximum weight matching maximizes the sum of the weights of its edges $e_{i\mapsto j}$



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 (possible) unmatched vertices, i.e., $n_s$ singletons $G_i$

# From the matching to the prolongator

We can formally define a *prolongator*:

$$P_c = \underbrace{\left[\begin{array}{cc} \underbrace{\left[\begin{array}{ccc} \mathbf{w}_{e_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{w}_{e_{n_p}} \end{array}\right]}_{n_p} 2n_p & \mathbf{0} \\ \mathbf{0} & \underbrace{\left[\begin{array}{ccc} w_1/|w_1| & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_{n_s}/|w_{n_s}| \end{array}\right]}_{n_s} n_s \end{array}\right]}_{n_c = n_p + n_s = J} \; n = 2n_p + n_s$$

$$= \begin{bmatrix} \tilde{P} & O \\ O & W \end{bmatrix} = [\mathbf{p}_1, \ldots, \mathbf{p}_J], \qquad \mathbf{w}_e = \frac{1}{\sqrt{w_i^2 + w_j^2}} \begin{bmatrix} w_i \\ w_j \end{bmatrix}$$

⇒ The $\mathcal{M}$ on (a log transformation of) $\hat{A}$ produces $A_f$ with dominant diagonal entries, for $P_f$ such that $P_c^T D P_f = 0$, with $D = \mathrm{diag}(A)$ (Olschowka et al. 1996, Duff et al., 2001)

# CMATCH (cont'd)

```
Input: $A$ matrix, $\mathbf{w}$ (smooth) vector,
maxsize maximum coarsest size
Output: hierarchy of coarse matrices $A^k$
1  $A^1 = A$, $k = 1$, $\mathbf{w^1} = \mathbf{w}$
2  while size($A^k$) $>$ maxsize
    2.1  apply parallel matching-based pairwise aggregation to the graph of $A^k$ with
         weigths depending on $\mathbf{w}^k$
    2.2  build $P_c^k$, $R_c^k = (P_c^k)^T$ and $A_c^k = R_c^k A^k P_c^k$
    2.3  $A^{k+1} = A_c^k$, $\mathbf{w}_c = R_c^k \mathbf{w}^k$
    2.4  $k = k + 1$
   endwhile
```

# CMATCH (cont'd)

**Input:** $A$ matrix, $\mathbf{w}$ (smooth) vector,
*maxsize* maximum coarsest size
**Output:** hierarchy of coarse matrices $A^k$

```
1 $A^1 = A$, $k = 1$, $\mathbf{w^1} = \mathbf{w}$
2 while size($A^k$) $>$ maxsize
    2.1 apply parallel matching-based pairwise aggregation to the graph of $A^k$ with weigths depending on $\mathbf{w}^k$
    2.2 build $P_c^k$, $R_c^k = (P_c^k)^T$ and $A_c^k = R_c^k A^k P_c^k$
    2.3 $A^{k+1} = A_c^k$, $\mathbf{w}_c = R_c^k \mathbf{w}^k$
    2.4 $k = k + 1$
  endwhile
```

## Increasing Coarsening Ratio for Reducing Complexity

Consecutive levels based on pairwise aggregation can be combined, e.g., double pairwise can be obtained by:

$$\overline{P_c}^k = P_c^{2k-1} P_c^{2k}, \quad \overline{R_c}^k = (\overline{P_c}^k)^T, \quad \overline{A_c}^k = A_c^{2k}, \quad k = 1, \dots \lceil nl/2 \rceil$$

# CMATCH (cont'd)

## Approximation matching algorithms & parallel software

efficient (sub-optimal) algorithms (Catalyürek et al. 2012, Manne et al. 2014)

- quality guarantee of the computed matching, generally $1/2-$approximation to a maximum weight matching
- linear-time $\mathcal{O}(nnz)$ complexity
- available software in source form (MatchBox-P by Halappanavar et al.)

# CMATCH (cont'd)

## Approximation matching algorithms & parallel software

efficient (sub-optimal) algorithms (Catalyürek et al. 2012, Manne et al. 2014)

- quality guarantee of the computed matching, generally $1/2-$approximation to a maximum weight matching
- linear-time $\mathcal{O}(nnz)$ complexity
- available software in source form (MatchBox-P by Halappanavar et al.)

## Main advantages of CMATCH

- a completely automatic procedure applicable to general s.p.d. systems, independent of any heuristics or a priori information on the near kernel of A

# CMATCH (cont'd)

## Approximation matching algorithms & parallel software

efficient (sub-optimal) algorithms (Catalyürek et al. 2012, Manne et al. 2014)

- quality guarantee of the computed matching, generally $1/2-$approximation to a maximum weight matching
- linear-time $\mathcal{O}(nnz)$ complexity
- available software in source form (MatchBox-P by Halappanavar et al.)

## Main advantages of CMATCH

- a completely automatic procedure applicable to general s.p.d. systems, independent of any heuristics or a priori information on the near kernel of A
- well-balanced coarse matrices among parallel processes, no need for special treatment of process-boundary dofs accounting for inter-processes coupling

# CMATCH (cont'd)

## Approximation matching algorithms & parallel software

efficient (sub-optimal) algorithms (Catalyürek et al. 2012, Manne et al. 2014)

- quality guarantee of the computed matching, generally $1/2-$approximation to a maximum weight matching
- linear-time $\mathcal{O}(nnz)$ complexity
- available software in source form (MatchBox-P by Halappanavar et al.)

## Main advantages of CMATCH

- a completely automatic procedure applicable to general s.p.d. systems, independent of any heuristics or a priori information on the near kernel of A
- well-balanced coarse matrices among parallel processes, no need for special treatment of process-boundary dofs accounting for inter-processes coupling
- significant flexibility in the choice of the size of aggregates, almost arbitrarily aggressive coarsening

# CMATCH (cont'd)

## Approximation matching algorithms & parallel software

efficient (sub-optimal) algorithms (Catalyürek et al. 2012, Manne et al. 2014)

- quality guarantee of the computed matching, generally $1/2-$approximation to a maximum weight matching
- linear-time $\mathcal{O}(nnz)$ complexity
- available software in source form (MatchBox-P by Halappanavar et al.)

## Main advantages of CMATCH

- a completely automatic procedure applicable to general s.p.d. systems, independent of any heuristics or a priori information on the near kernel of A
- well-balanced coarse matrices among parallel processes, no need for special treatment of process-boundary dofs accounting for inter-processes coupling
- possible improving in V-cycle convergence, by smoothing of matching-based prolongators as in classic smoothed aggregation

$$P_s^k = (I - \omega (D^k)^{-1} A^k) P_c^k, \text{ for } D^k = \mathrm{diag}(A^k)$$

# Quality and Convergence: *a posteriori* analysis

## Theorem

*(D'Ambra, Durastante, Filippone, Zikatanov, 2022) Our TG-AMG $B_{TG}$ with convergent smoother $M$ and BCMATCH exact algorithm has the following property:*

$$\|I - B_{TG}^{-1} A\|_A \leq 1 - \frac{\mu_c}{c^D}$$

$$\text{with} \quad \mu_c = \min_{1 \leq j \leq J} \mu_j(V_j^c) = \min_{1 \leq j \leq J} \left[ \max_{\mathbf{v}_j \in V_j} \min_{\mathbf{v}_j^c \in V_j^c} \frac{\|\mathbf{v}_j - \mathbf{v}_j^c\|_{D_j}^2}{\|\mathbf{v}_j\|_{A_j}^2} \right]$$

*and $c^D$ the continuity constant of the smoother.*

- The constants $c^D$ depends on the symmetrized $\tilde{M}$ convergent smoother

$$c_D \|\mathbf{v}\|_D^2 \leq \|\mathbf{v}\|_{\tilde{M}^{-1}}^2 \leq c^D \|\mathbf{v}\|_D^2$$

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

# Our recipies: highly parallel smoothers

Gauss-Seidel (GS): $A = M - N$, with $M = L + D$ and $N = -L^T$, where $D = \mathrm{diag}(A)$ and $L = \mathrm{tril}(A)$

It is intrinsically sequential!

# Our recipies: highly parallel smoothers

## Inexact block-Jacobi (HGS/$\ell_1$-HGS/HINVK)

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

# Our recipies: highly parallel smoothers

## Inexact block-Jacobi (HGS/$\ell_1$-HGS/HINVK)

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

$\ell_1$-HGS On process $p = 1, \ldots, np$ relative to the (row) index set $\Omega_p^{nb}$: $A_{pp} = L_{pp} + D_{pp} + L_{pp}^T$ where $D_{pp} = \mathrm{diag}(A_{pp})$ and $L_{pp} = \mathrm{tril}(A_{pp})$

# Our recipies: highly parallel smoothers

## Inexact block-Jacobi (HGS/$\ell_1$-HGS/HINVK)

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

$\ell_1$-HGS On process $p = 1, \ldots, np$ relative to the (row) index set $\Omega_p^{nb}$:
$A_{pp} = L_{pp} + D_{pp} + L_{pp}^T$ where $D_{pp} = \mathrm{diag}(A_{pp})$ and $L_{pp} = \mathrm{tril}(A_{pp})$

$$(M_{\ell_1-HGS})_p = L_{pp} + D_{pp} + D_{\ell_1 p}$$

$$D_{\ell_1 p} = diag((d_{\ell_1})_i)_{i=1,\ldots,nb}, \; d_{\ell_1} = \sum_{j \in \Omega_p^{nb}} |a_{ij}|$$

# Our recipies: highly parallel smoothers

## Inexact block-Jacobi (HGS/$\ell_1$-HGS/HINVK)

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

$\ell_1$-HGS On process $p = 1, \dots, np$ relative to the (row) index set $\Omega_p^{nb}$:
$A_{pp} = L_{pp} + D_{pp} + L_{pp}^T$ where $D_{pp} = \mathrm{diag}(A_{pp})$ and $L_{pp} = \mathrm{tril}(A_{pp})$

$$(M_{\ell_1-HGS})_p = L_{pp} + D_{pp} + D_{\ell_1 p}$$

$$D_{\ell_1 p} = diag((d_{\ell_1})_i)_{i=1,\dots,nb},\ d_{\ell_1} = \sum_{j \in \Omega_p^{nb}} |a_{ij}|$$

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

# Our recipies: highly parallel smoothers

## Inexact block-Jacobi (HGS/$\ell_1$-HGS/HINVK)

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

$\ell_1$-HGS On process $p = 1, \dots, np$ relative to the (row) index set $\Omega_p^{nb}$:
$A_{pp} = L_{pp} + D_{pp} + L_{pp}^T$ where $D_{pp} = \mathrm{diag}(A_{pp})$ and $L_{pp} = \mathrm{tril}(A_{pp})$

$$(M_{\ell_1-HGS})_p = L_{pp} + D_{pp} + D_{\ell_1 p}$$

$$D_{\ell_1 p} = diag((d_{\ell_1})_i)_{i=1,\dots,nb},\ d_{\ell_1} = \sum_{j \in \Omega_p^{nb}} |a_{ij}|$$

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

HINVK $M_{HINVK} = \mathrm{diag}((M_{HINVK})_p)_{p=1,\dots np}$

with $(M_{HINVK})_p = A_{pp}^{-1} \approx ZD^{-1}Z^T$ suitable for GPUs

# The weak approximation constant

Let be $P_c$ the CMATCH prolongator and $P_f$ its D-orthogonal complement, with $D = \mathrm{diag}(A)$. Our weak approximation constant is:

$$K = (\lambda_{\mathsf{min}}((P_f^T \tilde{M} P_f)^{-1}(P_f^T A P_f)))^{-1}$$

Unsmoothed prolongator with 3 sweeps of pairwise aggregation

| m | np | HGS | $\ell_1$−HGS | HINVK | $\ell_1$−INVK |
|---|---|---|---|---|---|
| 4096 | 1 | 1.3766 | 1.3766 | 1.5562 | 1.5562 |
| 2048 | 2 | 1.4194 | 1.5270 | 1.5273 | 1.7196 |
| 1024 | 4 | 1.4587 | 1.6621 | 1.6093 | 2.2149 |
| 512 | 8 | 1.4744 | 1.7803 | 1.8284 | 2.6713 |
| 256 | 16 | 1.4945 | 1.8230 | 1.8608 | 2.7307 |
| 128 | 32 | 1.5149 | 1.8682 | 1.8977 | 2.7972 |
| 64 | 64 | 1.5335 | 1.9162 | 1.9390 | 2.8715 |
| 32 | 128 | 1.5880 | 2.0343 | 2.0272 | 3.0707 |
| 16 | 256 | 1.6406 | 2.1594 | 2.1440 | 3.3688 |
| 8 | 512 | 1.6665 | 2.3088 | 2.3137 | 3.7280 |
| $l_1$-Jacobi | | | | 5.6220 | |

homogeneous 3D Laplacian problem with mesh size $m = 16^3$ over $np$ processes.

# Test Case: Poisson Equation

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

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

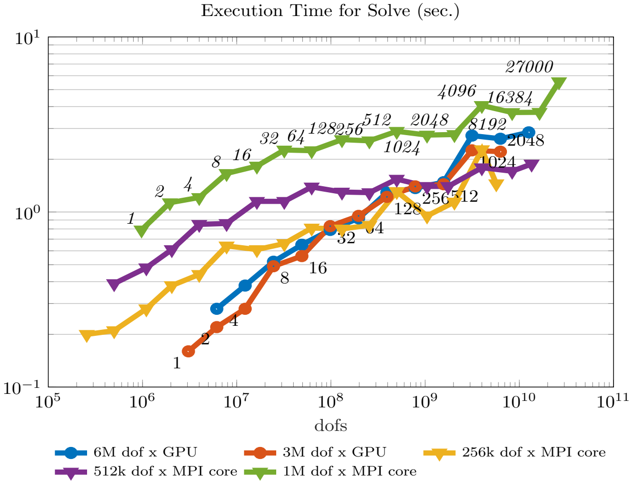
## Solver/preconditioner settings

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

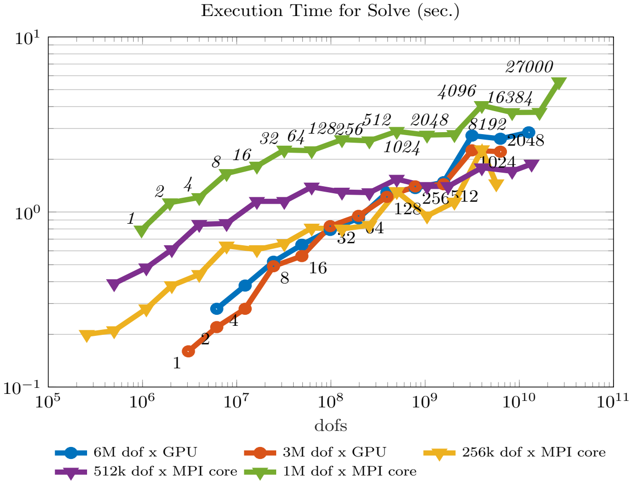
  VSCMATCH V-cycle, CMATCH building aggregates of max size $8$, smoothed prolongators
- coarsest matrix size $n_c \leq 200np$, with $np$ number of cores
- 1 sweep of forward/backward Hybrid Gauss-Seidel smoother (4 sweeps of $l_1-$Jacobi on GPU), parallel PCG coupled with Block-Jacobi+ILU(0) at the coarsest 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

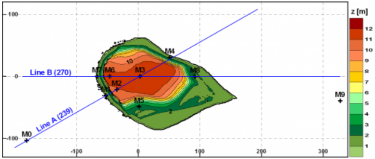


## Performance/Power efficiency

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

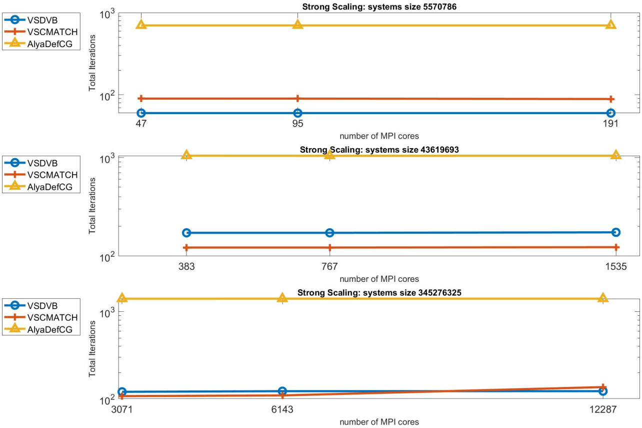


Bolund is an isolated hill situated in Roskilde Fjord, Denmark. An almost vertical escarpment in the prevailing W-SW sector ensures flow separation in the windward edge resulting in a complex flow field.

- **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 - Strong Scaling - Pressure Equation

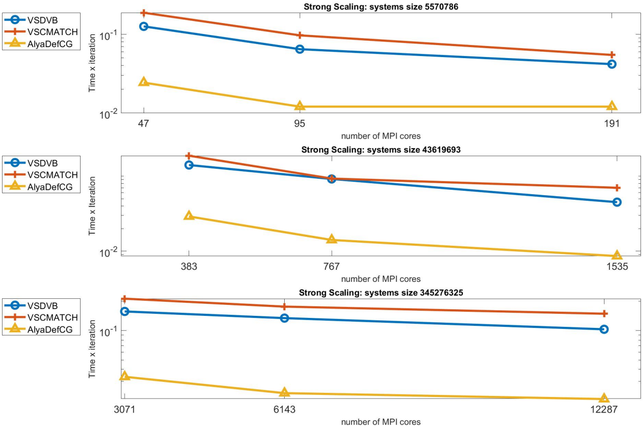
Three fixed size problems ($\approx 6 \times 10^6,\ 4.4 \times 10^7,\ 0.35 \times 10^9$), for increasing number of cores, 20 time steps in the fully development flow phase



- AMG preconditioners largely reduce the total number of iterations

# Bolund Test Case - Strong Scaling - Pressure Equation

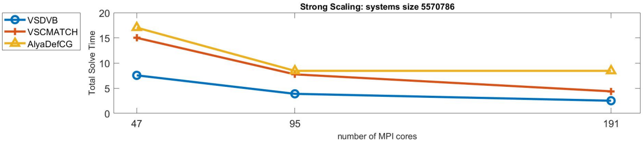
Three fixed size problems ($\approx 6 \times 10^6,\ 4.4 \times 10^7,\ 0.35 \times 10^9$), for increasing number of cores, 20 time steps in the fully development flow phase

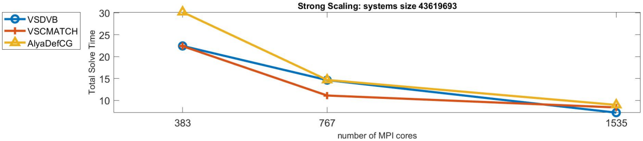


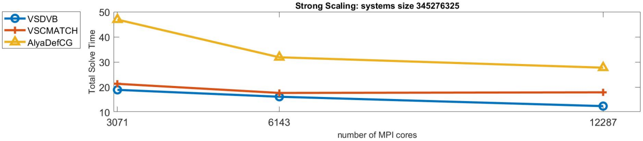
- solve time needed per each iteration decreases for increasing number of cores

# Bolund Test Case - Strong Scaling - Pressure Equation

Three fixed size problems ($\approx 6 \times 10^6,\ 4.4 \times 10^7,\ 0.35 \times 10^9$), for increasing number of cores, 20 time steps in the fully development flow phase







- the trade-off between cost-per-iteration and number of iterations advantages the AMG preconditioners

# 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 coarsening algorithm based on compatible weighted matching, used in conjunction with smoothed prolongators and highly parallel smoothers, shows algorithmic and implementation scalability
- we solve systems with size larger than $10^{10}$ on current pre-exascale computers, embedding hybrid CPU-GPU nodes, saving time and energy
- scalability results and comparison with available software demonstrates the validity of our approaches both in terms of algorithms and in terms of software development
- 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 $(10^{18})$ of dofs

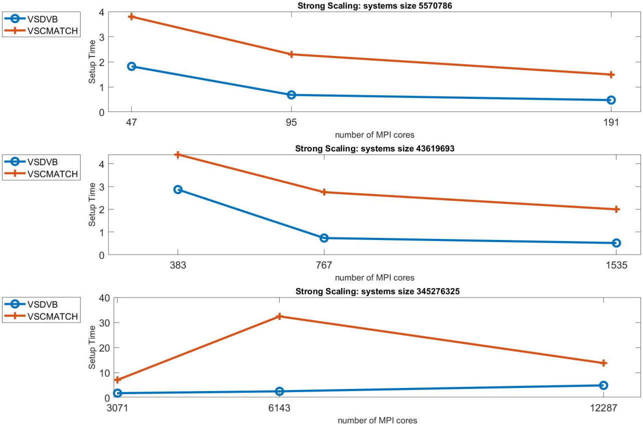
# Main recent references

1. P. D'Ambra, F. Durastante, S. Filippone, L. Zikatanov, On the Quality of Matching-based Aggregates for Algebraic Coarsening of SPD Matrices in AMG, 2022. Under revision.
2. D. Bertaccini, P. D'Ambra, F. Durastante, S. Filippone, Preconditioning Richards Equations: Spectral Analysis and Parallel Solution at Very Large Scale, 2021. Under revision.
3. 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.
4. H. Owen, G. Houzeaux, F. Durastante, S. Filippone, P. D'Ambra, AMG4PSBLAS Linear Algebra Package brings Alya one step closer to Exascale, in Proceedings of 32nd International Conference on Parallel Computational Fluid Dynamics (ParCFD) 2021.
5. M. Bernaschi, P. D'Ambra, D. Pasquini, AMG based on Compatible Weighted Matching on GPUs, Parallel Computing. Vol. 92, 2020.

## 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

# Preconditioners Setup Time in Alya


Strong Scaling: systems size 5570786
VSDVB
VSCMATCH
Setup Time
number of MPI cores
Strong Scaling: systems size 43619693
VSDVB
VSCMATCH
Setup Time
number of MPI cores
Strong Scaling: systems size 345276325
VSDVB
VSCMATCH
Setup Time
number of MPI cores