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

Pasqua D'Ambra

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

CaSToRC Summer 2021 Online Seminar Series

13 July, 2021

The HPC Team at IAC-CNR

Co-authors of this work :

- Fabio Durastante (Univ. of Pisa and IAC-CNR), IT
- Salvatore Filippone (Univ. of Rome Tor-Vergata and IAC-CNR), IT

Projects collaborators :

- Massimo Bernaschi (IAC-CNR), IT
- Daniele Bertaccini (Univ. of Rome Tor-Vergata and IAC-CNR), IT
- Mauro Carrozzo (IAC-CNR), IT
- Dario Pasquini (IAC-CNR), IT
- Gabriele Salvati (IAC-CNR), IT

Greetings :

Panayot S. Vassilevski, CASC-LLNL (Livermore, CA) and Portland State University (Portland, OR), USA

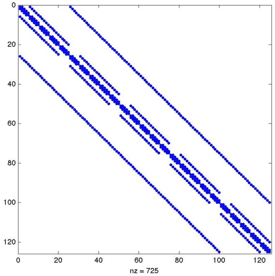
Mahantesh M. Halappanavar, PNNL (Richland, WA), USA

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



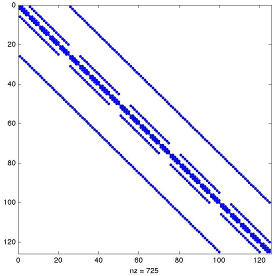
often the most time consuming computational kernel in many areas of Computational/Data Science

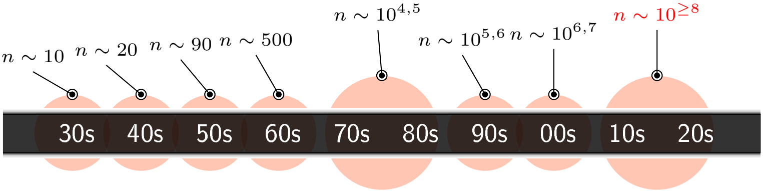
What we want to solve

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

n **large**

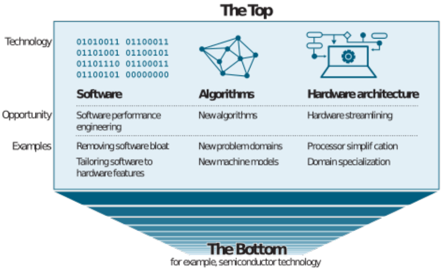
$$\text{sparsity degree} = 1 - \frac{nnz}{n^2} \approx 1$$





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 of unknowns

The driving philosophy



The Top

	Software	Algorithms	Hardware architecture
Technology			
Opportunity	Software performance engineering	New algorithms	Hardware streamlining
Examples	Removing software bloat Tailoring software to hardware features	New problem domains New machine models	Processor simplif cation Domain specialization

The Bottom

for example, semiconductor technology

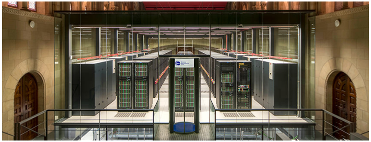
C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, and T. B. Schardl, "There's plenty of room at the Top: What will drive computer performance after Moore's law?", Science (2020)

"As miniaturization wanes, the silicon-fabrication improvements at the Bottom will no longer provide the predictable, broad-based gains in computer performance that society has enjoyed for more than 50 years. Software performance engineering, development of algorithms, and hardware streamlining at the Top can continue to make computer applications faster in the post-Moore era"

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
⋮	⋮	⋮	⋮
14	Marconi-100	347,776	21,640.0
15	Piz Daint	387,872	21,230.0
⋮	⋮	⋮	⋮
63	MareNostrum	153,216	6,470.8

- Computers with thousands of MPI cores
- Hybrid form of parallelism: MPI, OpenMP, CUDA/OpenCL/OpenACC, . . .



MareNostrum IV - BSC

Piz Daint - CSCS

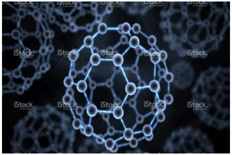
[1] TOP500 list, June 2021 – https://www.top500.org

Context:EoCoE-II 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-II target applications

Wind Models

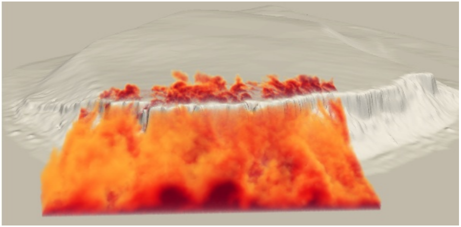
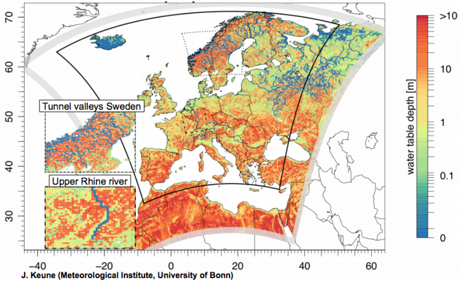


Image credits H. Owen and G. Marin, Barcelona Supercomputing Centre

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

Regional Hydrological Models



J. Keune (Meteorological Institute, University of Bonn)

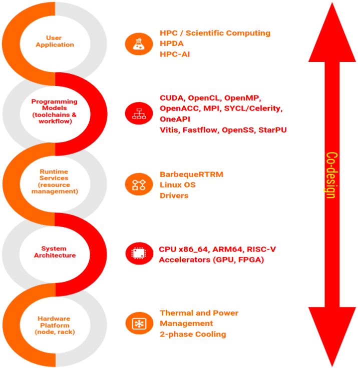
- Darcy equation,
- Richards equation,
- Equations for overland flow
- . . .

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

Context: 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)

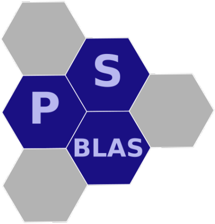
Hardware Platform (node, rack)

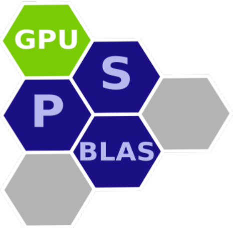
Thermal and Power
Management
2-phase Cooling

Co-design

Our contribution: performance/power efficient MathLib

Parallel Sparse Computation Toolkit

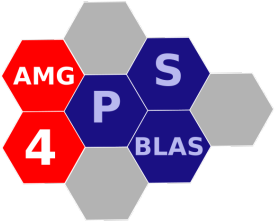




Parallel Sparse Basic Linear Algebra Subroutines

+

its GPU-plugin



AMG Preconditioners for PSBLAS

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

Recently selected as Excellent Innovation from EU Innovation Radar

PSBLAS

A Software development project started in the early 2000s, prompted by the work of Iain Duff et al. on standard for Sparse BLAS, ACM TOMS 23 (1997)

- parallel sparse matrix operations and data structures management, Krylov solvers (for spd and general matrices)
- general row-block matrix distribution, support infrastructure for mesh handling and sparse matrix I/O
- data allocation through graph partitioning (METIS, ParMETIS, SCOTCH)
- object oriented design in Fortran 2003/2008

S. Filippone et al., PSBLAS: A library for parallel linear algebra computation on sparse matrices, ACM TOMS, 26, 4, 2000.

S. Filippone et al., Object-Oriented Techniques for Sparse Matrix Computations in Fortran 2003. ACM TOMS, 38, 4, 2012.

PSBLAS (contd.)

- message-passing paradigm (MPI), CUDA plugin
- internal matrix representation/storage: distributed sparse matrix with native CSR/CSC/COO format, extension plugin for many other storage formats, including CUDA-enabled (DIAG, ELLPACK and variations)
- multiple tools for storage transformations
- tools and data structures for global/local index mapping (long/short integers for global/local numbering handled separately) and (halo data exchange);

PSBLAS (contd.)

- message-passing paradigm (MPI), CUDA plugin
- internal matrix representation/storage: distributed sparse matrix with native CSR/CSC/COO format, extension plugin for many other storage formats, including CUDA-enabled (DIAG, ELLPACK and variations)
- multiple tools for storage transformations
- tools and data structures for global/local index mapping (long/short integers for global/local numbering handled separately) and (halo data exchange);

Memory footprint (main issue for scalability)

- matrices: proportional to number of local rows/indices (may require padding for vectorization on GPUs and similar)
- vectors: proportional to local indices plus halo indices
- data exchange auxiliary storage: proportional to number of boundary plus halo indices
- global/local index mappings: proportional to local plus halo indices (may trade more memory for speed if acceptable)

AMG4PSBLAS: AMG Preconditioners for PSBLAS

A software development project started in 2004

- initially developed as a package of algebraic multigrid Schwarz preconditioners, extended to more general AMG preconditioning
- 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

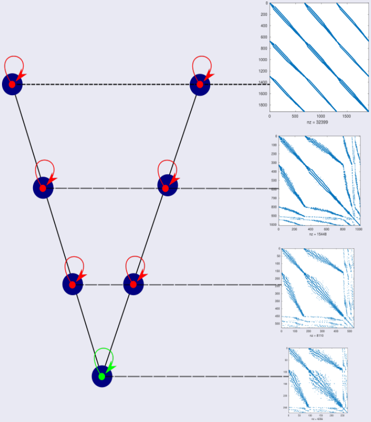
P. D'Ambra et al., MLD2P4: a Package of Parallel Algebraic Multilevel Domain Decomposition Preconditioners in Fortran 95, ACM TOMS, 37, 3, 2010

P. D'Ambra et al., AMG preconditioners for Linear Solvers towards Extreme Scale. Preprint arXiv:2006.16147 (to appear on SISC, 2021)

AMG Preconditioners

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)

Scalable (optimal) preconditioners

Solve the system:

$$BAx = Bb,$$

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

- $\mu(BA) \approx 1$, 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 (optimal) preconditioners

Solve the system:

$$BAx = Bb,$$

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

- $\mu(BA) \approx 1$, 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

AMG4PSBLAS methods

AMG4PSBLAS preconditioners can be obtained as combination of

setup or coarsening phase: parallel decoupled/coupled

- DVB smoothed aggregation based on the usual strength of connection measure (Vaněk and Brezina, 1996)
- CMATCH smoothed and unsmoothed aggregation based on compatible weighted matching (D'Ambra et al., 2013, 2016, 2018, 2021)

solve phase: **available on GPU** for many choices of smoothers & coarsest solver

- cycles V, W, K
- smoothers many versions of inexact block-Jacobi, Hybrid (F/B) Gauss-Seidel, additive Schwarz
- coarsest-solvers parallel sparse LU (MUMPS, SuperLU, UMFPACK), many versions of inexact block-Jacobi, hybrid (F/B) Gauss-Seidel, iterative preconditioned Krylov solvers

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!

Highly Parallel Smoothers

Inexact block-Jacobi (HGS/ℓ_1-HGS/AINV)

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

Highly Parallel Smoothers

Inexact block-Jacobi (HGS/ℓ_1-HGS/AINV)

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

Highly Parallel Smoothers

Inexact block-Jacobi (HGS/ℓ_1-HGS/AINV)

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

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

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

Highly Parallel Smoothers

Inexact block-Jacobi (HGS/ℓ_1-HGS/AINV)

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

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

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

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

Highly Parallel Smoothers

Inexact block-Jacobi (HGS/ℓ_1-HGS/AINV)

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

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

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

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

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

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

Parallel coarsening based on compatible weighted matching (CMATCH)

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

Parallel coarsening based on compatible weighted matching (CMATCH)

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 (Brandt, 2000; Falgout and Vassilevski 2004)

$A_f = P_f^T A P_f$ as well conditioned as possible,

i.e., convergence rate of compatible relaxation $\rho_f = \|I - M_f^{-1} A_f\|_{A_f} << 1$

Parallel coarsening based on compatible weighted matching (CMATCH)

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 (Brandt, 2000; Falgout and Vassilevski 2004)

$A_f = P_f^T A P_f$ as well conditioned as possible,

i.e., convergence rate of compatible relaxation $\rho_f = \|I - M_f^{-1} A_f\|_{A_f} << 1$

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**

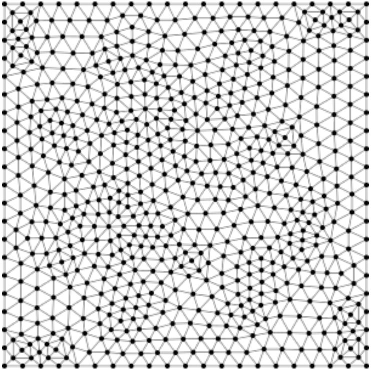
CMATCH (cont'd)

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 $e_{i \mapsto j}$



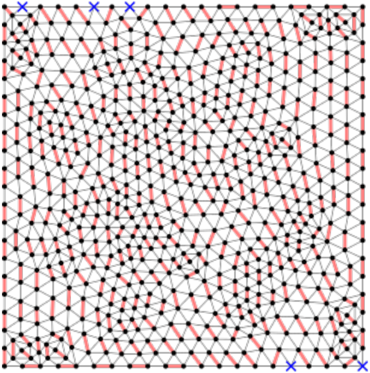
CMATCH (cont'd)

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*:

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

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

⇒ The $\mathcal{M}$ on (a log transformation of) $\hat{A}$ produces A_f with dominant diagonal entries (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:

$$\overline{P}^k = P^{2k-1} P^{2k}, \quad \overline{R}^k = (\overline{P}^k)^T, \quad \overline{A}^k = A^{2k}, \quad k = 1, \ldots \lceil nl/2 \rceil$$

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

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

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

KCMATCH	K-cycle with 2 inner iterations, CMATCH building aggregates of max size 8, unsmoothed prolongators
VSCMATCH	V-cycle, CMATCH building aggregates of max size 8, smoothed prolongators
VSDVB	V-cycle for decoupled classic smoothed aggregation

- coarsest matrix size $n_c \leq 200np$, with np number of cores
- 1 sweep of forward/backward Hybrid Gauss-Seidel smoother, parallel CG preconditioned with Block-Jacobi and ILU(0) at the coarsest level

Experimental environment & Comparison

Piz Daint - Swiss National Supercomputing Center by PRACE

- Cray Model XC40/Cray XC50 architecture with 5704 hybrid compute nodes (Intel Xeon E5-2690 v3 with Nvidia Tesla P100 accelerator)
- Cray Aries routing and communications ASIC with Dragonfly network topology
- GNU compiler rel. 8, Cray MPI 7, Cray-libsci 20.09.1
- PSBLAS 3.7, AMG4PSBLAS 1.0

Hypre: Scalable Linear Solvers and Multigrid Methods by LLNL

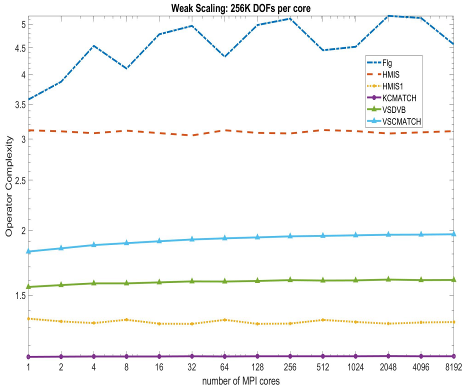
- BoomerAMG as preconditioner of CG, stopped when $\|\mathbf{r}^k\|_2/\|\mathbf{b}\|_2 \leq 10^{-6}$, or $itmax = 500$
- V-cycle with 1 sweep of forward/backward Hybrid Gauss-Seidel smoother, LU factorization at the coarsest level
- 3 coarsening schemes: hybrid RS/CLJP (**Flg**), Hybrid Maximal Independent Set (**HMIS**), HMIS with first level of aggressive coarsening (**HMIS1**); default parameters for coarsest matrix size $1 \leq n_c \leq 9$, coupled with modified (long-range) classical interpolation

Weak scaling (256K dofs per core): algorithmic scalability

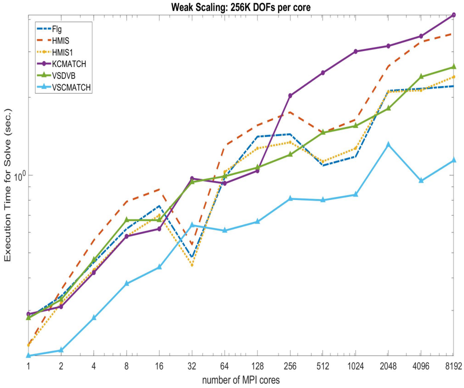
		AMG4PSBLAS			Hypre		
np	$n/10^6$	KCMATCH	VSCMATCH	VSDVB	Flg	HMIS	HMIS1
1	0.256	12	7	11	6	6	12
2	0.512	12	7	12	7	9	15
2^2	1.036	12	7	13	7	12	17
2^3	2.048	12	7	14	8	13	17
2^4	4.075	12	8	14	8	14	20
2^5	8.049	13	9	15	8	14	20
2^6	16.384	12	8	15	9	16	22
2^7	32.604	12	8	15	10	18	25
2^8	63.917	13	9	16	10	20	27
2^9	131,072	14	8	18	11	22	29
2^{10}	256,000	15	8	17	12	25	32
2^{11}	511,335	16	12	21	13	29	37
2^{12}	1024,192	15	8	26	13	35	40
2^{13}	2097,152	16	9	27	14	37	44

Table: Number of iterations for solve

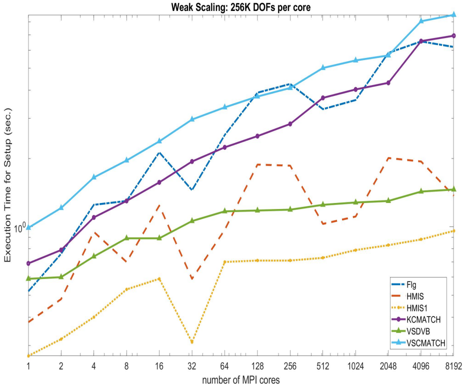
Weak scaling (256K dofs per core): operator complexity



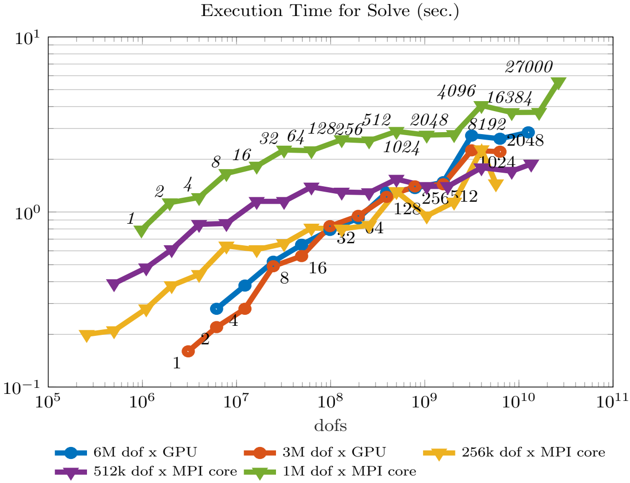
Weak scaling (256K dofs per core): solve time



Weak scaling (256K dofs per core): setup time

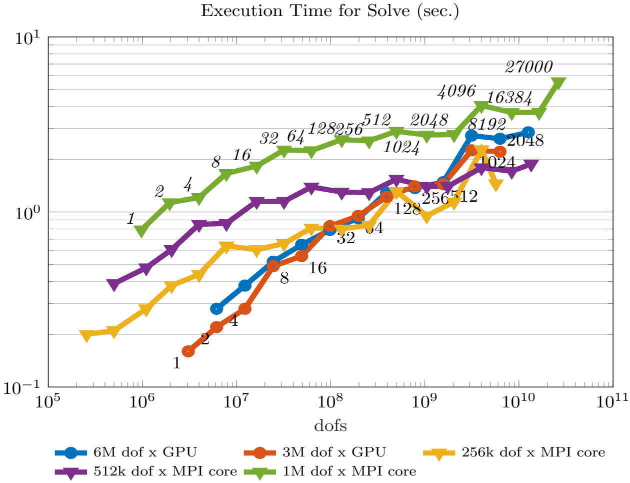


Results at extreme scale: MPI vs hybrid MPI-CUDA[2]



[2] on GPUs 1 sweep of ℓ_1-Jacobi smoother is used.

Results at extreme scale: MPI vs hybrid MPI-CUDA



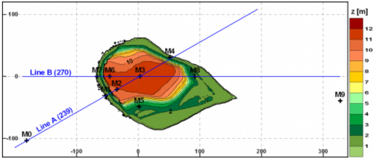
Performance/Power efficiency

the hybrid approach permits savings in solve time and energy consumption

A CFD application inside Alya

BSC

Joint work with
Herbert Owen
Barcelona Super Computing Center

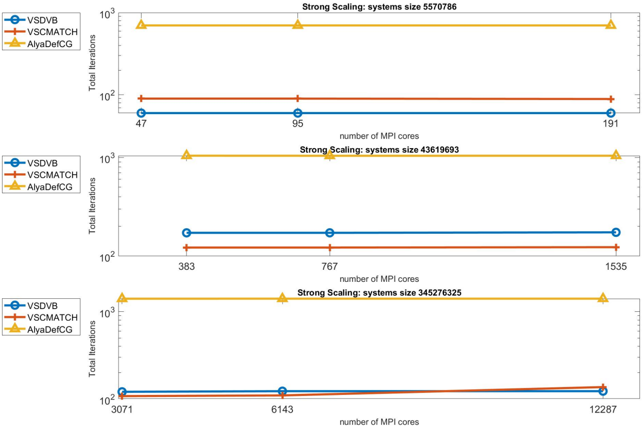


Bolund is an isolated hill situated in Roskilde Fjord, Denmark. An almost vertical escarpment in the prevailing W-SW sector ensures flow separation in the windward edge resulting in a complex flow field.

- **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

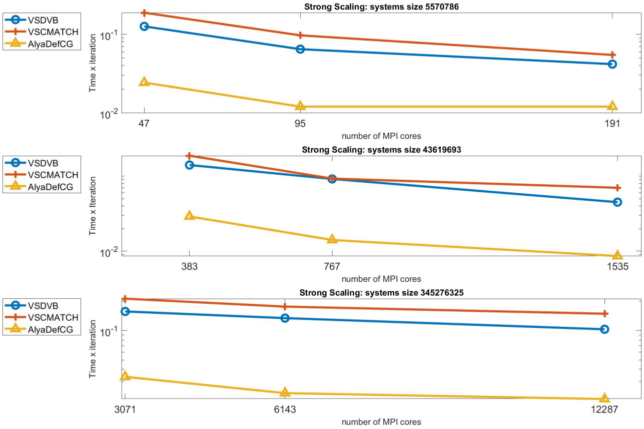
Three fixed size problems ($\approx 6 \times 10^6,\ 4.4 \times 10^7,\ 0.35 \times 10^9$), for increasing number of cores, 20 time steps in the fully development flow phase



- AMG preconditioners largely reduce the total number of iterations

Bolund Test Case - Strong Scaling - Pressure Equation

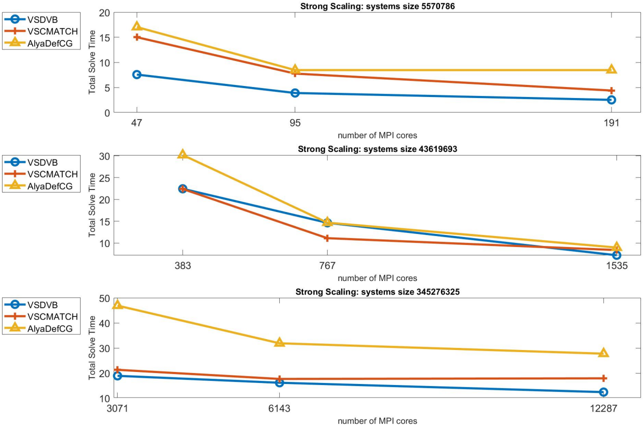
Three fixed size problems ($\approx 6 \times 10^6,\ 4.4 \times 10^7,\ 0.35 \times 10^9$), for increasing number of cores, 20 time steps in the fully development flow phase



- 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 new software framework addressing scalability, flexibility and robusteness for high-performance scientific computing at extreme scale
- the 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, in less than 3 seconds
- 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 hydrologycal applications, in collaborations with BSC and JSC, is work in progress

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

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 2001.09969 preprint (2020).
- 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 2006.16147 preprint.
- 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.