On highly scalable implicit solvers for multiphysics

Santiago Badia

TEAM: J. Príncipe, A. Martín, R. Planas, O. Colomés, A. Hierro, M. Olm, H. Nguyen, J. Bonilla

HPSC Team, Centre Internacional de Mètodes Numèrics a l'Enginyeria (CIMNE)
Castelldefels, Spain

Universitat Politècnica de Catalunya
Barcelona, Spain

RES Engineering Seminar, Barcelona, October 2014

Introduction

Part I: **Scalable solvers**

- This work grounds on our recent efforts*,† towards the development of highly scalable domain decomposition linear solvers for FE analysis
- These codes rely on a novel implementation of Balancing Domain Decomposition by Constraints (BDDC) preconditioning
- Scalability systematically assessed for 3D elliptic PDEs (Poisson, Elasticity) with remarkable results (e.g., weakly scales up to $>$ 370K IBM BG/Q cores)

* S. Badia, A. F. Martín and J. Principe. A highly scalable parallel implementation of balancing domain decomposition by constraints. *SIAM J. Sci. Comput.* Vol. 36(2), pp. C190-C218, 2014.

† S. Badia, A. F. Martín and J. Principe. On the scalability of inexact balancing domain decomposition by constraints with overlapped coarse/fine corrections. Submitted, 2014.

Introduction

Part II: **Multiphysics solvers**

- Final goal is extreme-scale multiphysics solvers based on *recursive* block-preconditioning*, where highly scalable one-physics solvers are the building blocks

- In the road to more complex problems, some experiences with BDDC-based parallel solvers for incompressible flows† (continuous pressure spaces)

* S. Badia, A. F. Martín and R. Planas. Block recursive LU preconditioners for the thermally coupled incompressible inductionless MHD problem. *Journal of Computational Physics*, Vol. 274, pp. 562-591, 2014.

† S. Badia, and A. F. Martín. Balancing domain decomposition preconditioning for the discrete Stokes problem with continuous pressures. In preparation, 2014.

Part I

Highly scalable solvers

Part I: Highly scalable solvers

1. BDDC preconditioner
2. Highly scalable implementation
3. Inexact BDDC
4. Multilevel BDDC

Outline

Domain Decomposition

- Let us consider a symmetric, coercive problem (e.g., thermal or elasticity problem) in Ω
- We can approximate the problem using Finite Elements, via a triangulation $\mathcal{T}(\Omega)$
- **Algebraic problem**: Find

$$x \in \mathbb{R}^n \ : \ Ax = b,$$

A is a large, sparse, and symmetric positive definite (also for nonsym.)

Domain Decomposition

- Let us consider a symmetric, coercive problem (e.g., thermal or elasticity problem) in Ω
- We can approximate the problem using Finite Elements, via a triangulation $\mathcal{T}(\Omega)$
- **Algebraic problem**: Find

$$x \in \mathbb{R}^n \; : \; Ax = b,$$

A is a large, sparse, and symmetric positive definite (also for nonsym.)

Domain Decomposition

- Let us consider a symmetric, coercive problem (e.g., thermal or elasticity problem) in Ω
- We can approximate the problem using Finite Elements, via a triangulation $\mathcal{T}(\Omega)$
- **Algebraic problem**: Find

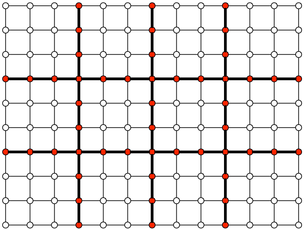
$$x \in \mathbb{R}^n \ : \ Ax = b,$$

A is a large, sparse, and symmetric positive definite (also for nonsym.)

Motivation:

Efficient exploitation of distributed-memory machines for large scale FE problems $\Rightarrow$ **Domain decomposition framework**

○: interior DoFs (I); ●: interface dofs (Γ)



Preconditioned iterative solvers

Preconditioned iterative solvers are the only *scalable* choice on ($>$ 100Kcores)

- `matvec` and `aplyprec` per iteration
- Key ingredient: preconditioner M^{-1}
- E.g., $M^{-1} = A^{-1}$, sol'on in 1 iteration
- Weak scaling (facing ever-increasing scales)
- No preconditioning: blow-up iterations!
- Local preconditioners (NN) without global coupling idem

PCG ($Ax = f$)

$r_0 := f - Ax_0$
$z_0 := M^{-1} r_0$
$p_0 := z_0$
for $j = 0, \ldots,$ till CONV **do**
    $s_{j+1} = Ap_j$
    $\ldots$
    $z_{j+1} := M^{-1} r_{j+1}$
    $\ldots$
end for

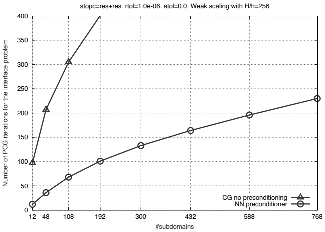
Preconditioned iterative solvers

Preconditioned iterative solvers are the only *scalable* choice on ($>$ 100Kcores)

- `matvec` and `aplyprec` per iteration
- Key ingredient: preconditioner M^{-1}
- E.g., $M^{-1} = A^{-1}$, sol'on in 1 iteration
- Weak scaling (facing ever-increasing scales)
- No preconditioning: blow-up iterations!
- Local preconditioners (NN) without global coupling idem

PCG ($Ax = f$)

$r_0 := f - Ax_0$
$z_0 := M^{-1} r_0$
$p_0 := z_0$
for $j = 0, \dots,$ till CONV **do**
  $s_{j+1} = Ap_j$
  $\dots$
  $z_{j+1} := M^{-1} r_{j+1}$
  $\dots$
end for



fixed N/P with P ↑

BDDC Balancing DD by constraints

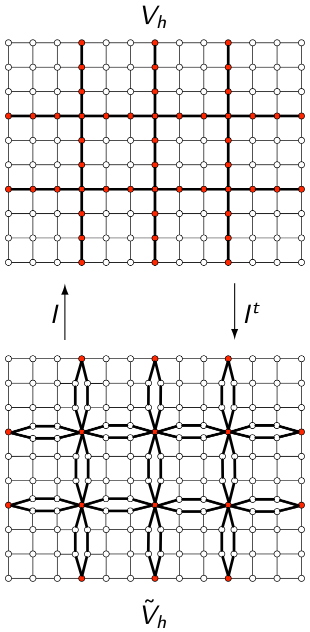
Idea: Solve problem w/ reduced continuity

- Find $\tilde{x} \in \mathbb{R}^{\tilde{n}}$ such that:

$$\tilde{A}\tilde{x} = I^t r$$

  and obtain $z = M_{BDDC} r = \mathcal{E} I \tilde{x}$

- $\tilde{A}$ is a sub-assembled global matrix (only assembled the red corners)
- $I : \tilde{V}_h \longrightarrow V_h$ is an injection (weight, comm and add)
- $\mathcal{E}$ is the harmonic extension operator (local problems to make interior residual zero)

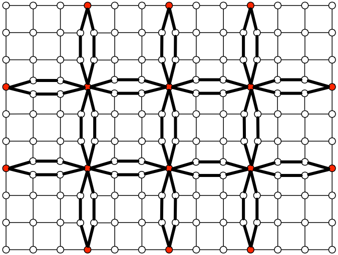


BDDC preconditioning

- Let $\tilde{V}_h = [\tilde{v}_{\circ} \ \tilde{v}_{\bullet}]$ and decompose $\tilde{V}_h$ as

$$\tilde{V}_h = \tilde{V}_F \oplus \tilde{V}_C, \text{ with } \begin{cases} \tilde{V}_F = [\tilde{v}_{\circ} \ 0] \\ \tilde{V}_C \perp_{\tilde{A}} \tilde{V}_F \end{cases}$$

- Now, problem split into fine-grid ($\tilde{x}_F$) and coarse-grid ($\tilde{x}_C$) correction



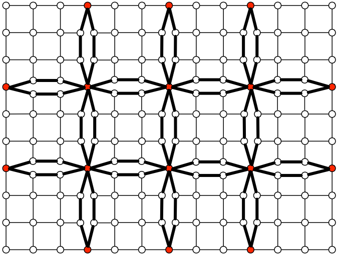
$\tilde{V}_h$

BDDC preconditioning

- Let $\tilde{V}_h = [\tilde{v}_{\circ} \ \tilde{v}_{\bullet}]$ and decompose $\tilde{V}_h$ as

$$\tilde{V}_h = \tilde{V}_F \oplus \tilde{V}_C, \text{ with} \begin{cases} \tilde{V}_F = [\tilde{v}_{\circ} \ 0] \\ \tilde{V}_C \perp_{\tilde{A}} \tilde{V}_F \end{cases}$$

- Now, problem split into fine-grid ($\tilde{x}_F$) and coarse-grid ($\tilde{x}_C$) correction



$\tilde{V}_h$

HPSC High Performance Scientific Computing

CIMNE

BDDC preconditioning

- Let $\tilde{V}_h = [\tilde{v}_\circ \ \tilde{v}_\bullet]$ and decompose $\tilde{V}_h$ as

$$\tilde{V}_h = \tilde{V}_F \oplus \tilde{V}_C, \text{ with} \begin{cases} \tilde{V}_F = [\tilde{v}_\circ \ 0] \\ \tilde{V}_C \perp_{\tilde{A}} \tilde{V}_F \end{cases}$$

- Now, problem split into fine-grid $(\tilde{x}_F)$ and coarse-grid $(\tilde{x}_C)$ correction

Fine-grid correction $(\tilde{x}_F)$

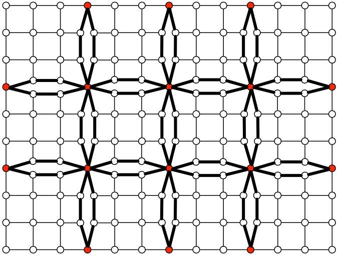
- Find $\tilde{x}_F \in \mathbb{R}^{\tilde{n}}$ such that

$$\tilde{A}\tilde{x}_F = I^t r, \text{ constrained to } (\tilde{x}_F)_\bullet = 0$$

- Equivalent to P independent problems

Find $\tilde{x}_F^{(i)} \in \mathbb{R}^{\tilde{n}^{(i)}}$ such that

$$A^{(i)}\tilde{x}_F^{(i)} = I_i^t r, \text{ constrained to } (\tilde{x}_F^{(i)})_\bullet = 0$$



$\tilde{V}_h$

BDDC preconditioning

- Let $\tilde{V}_h = [\tilde{v}_\circ \ \tilde{v}_\bullet]$ and decompose $\tilde{V}_h$ as

$$\tilde{V}_h = \tilde{V}_F \oplus \tilde{V}_C, \ \text{with} \begin{cases} \tilde{V}_F = [\tilde{v}_\circ \ 0] \\ \tilde{V}_C \perp_{\tilde{A}} \tilde{V}_F \end{cases}$$

- Now, problem split into fine-grid ($\tilde{x}_F$) and coarse-grid ($\tilde{x}_C$) correction

Coarse-grid correction ($\tilde{x}_C$)

Computation of $\tilde{V}_C = \mathrm{span}\{\Phi_1, \Phi_2, \ldots, \Phi_{n_C}\}$

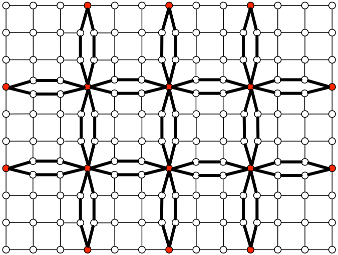
- Find $\Phi \in \mathbb{R}^{\tilde{n} \times n_C}$ such that

$$\tilde{A}\tilde{\Phi} = 0, \ \text{constrained to } \Phi_\bullet = I$$

- Equivalent to P independent problems

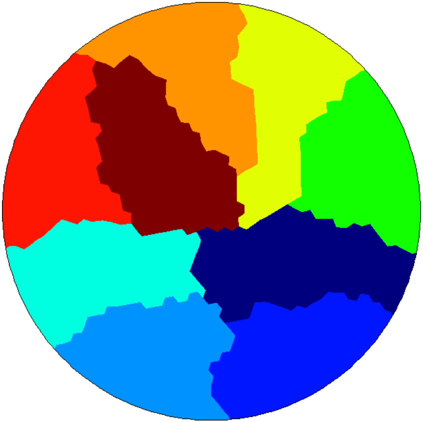
Find $\Phi^{(i)} \in \mathbb{R}^{\tilde{n} \times n_C^{(i)}}$ such that

$$A^{(i)}\Phi^{(i)} = 0, \ \text{constrained to } \Phi_\bullet^{(i)} = I$$

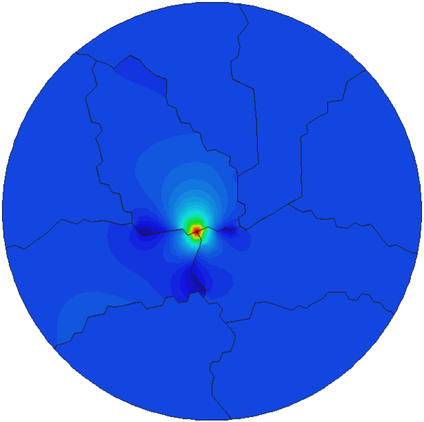


$\tilde{V}_h$

BDDC coarse corner function



Circle domain partitioned into 9 subdomains



Φ_j ($\tilde{V}_C$'s basis vector)

BDDC preconditioning

- Let $\tilde{V}_h = [\tilde{v}_{\circ} \ \tilde{v}_{\bullet}]$ and decompose $\tilde{V}_h$ as

$$\tilde{V}_h = \tilde{V}_F \oplus \tilde{V}_C, \text{ with} \begin{cases} \tilde{V}_F = [\tilde{v}_{\circ} \ 0] \\ \tilde{V}_C \perp_{\tilde{A}} \tilde{V}_F \end{cases}$$

- Now, problem split into fine-grid $(\tilde{x}_F)$ and coarse-grid $(\tilde{x}_C)$ correction

Coarse-grid correction $(\tilde{x}_C)$

Assembly and solution of coarse-grid problem

$$A_C = \text{assembly}(\Phi^t A^{(i)} \Phi), \quad \text{Solve } A_C \alpha_c = \Phi^t I^t r, \quad \tilde{x}_C = \Phi \alpha_C$$

Coarse-grid problem is

- Global, i.e. couples all subdomains
- But much smaller than original Schur complement S (size $\mathrm{n_C}$)
- Potential loss of parallel efficiency with P

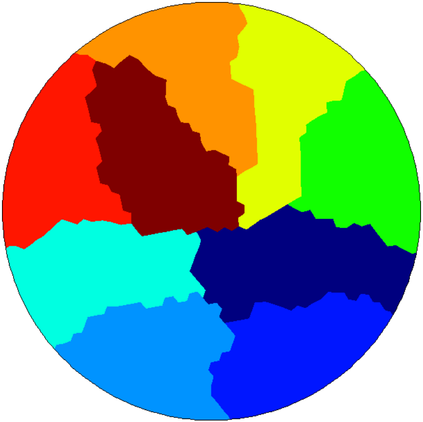
Coarse dof's definition

Key aspect: Selection of coarse dofs, i.e. continuity among subdomains

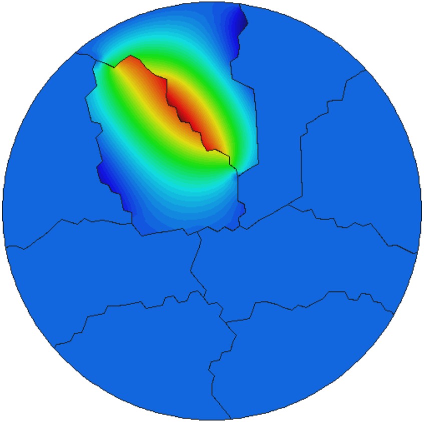
- Weak scalability ($\kappa(M_{BDDC}A)$ *constant* for fixed N/P and $\uparrow P$)
- N/P large in practice $\sim \mathcal{O}(10^{4-5})$
- BDDC(ce) and BDDC(cef) require much less iterations in $3D$
- But at the expense of a more costly coarse-grid problem

Coarse dofs vs. $\kappa(M_{BDDC}A)$:	$d = 2$	$d = 3$
Continuity on corners	$\left[1 + d^{-1}\log^2\left(\frac{N}{P}\right)\right]$	$\frac{N}{P}\left[1 + d^{-1}\log^2\left(\frac{N}{P}\right)\right]$
Continuity of mean value on edges too	$\left[1 + d^{-1}\log^2\left(\frac{N}{P}\right)\right]$	$\left[1 + d^{-1}\log^2\left(\frac{N}{P}\right)\right]$
Continuity of mean value on faces too	-	$\left[1 + d^{-1}\log^2\left(\frac{N}{P}\right)\right]$

BDDC coarse edge function



Circle domain partitioned into 9 subdomains



Φ_j ($\tilde{V}_C$'s basis vector)

HPSC
High Performance Scientific Computing

Outline

CIMNE

Why BDDC for extreme scales?

1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can be solved **inexactly** (AMG-cycle)
5. A **multilevel** extension is possible (for extreme core counts)

Why BDDC for extreme scales?

1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can be solved **inexactly** (AMG-cycle)
5. A **multilevel** extension is possible (for extreme core counts)

Why BDDC for extreme scales?

1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can be solved **inexactly** (AMG-cycle)
5. A **multilevel** extension is possible (for extreme core counts)

Why BDDC for extreme scales?

1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can be solved **inexactly** (AMG-cycle)
5. A **multilevel** extension is possible (for extreme core counts)

Why BDDC for extreme scales?

1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can be solved **inexactly** (AMG-cycle)
5. A **multilevel** extension is possible (for extreme core counts)

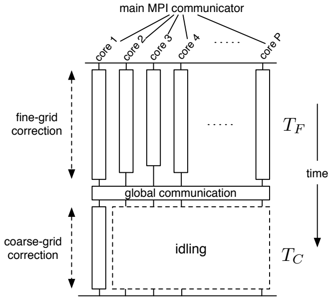
Why BDDC for extreme scales?

1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can be solved **inexactly** (AMG-cycle)
5. A **multilevel** extension is possible (for extreme core counts)

- (1)-(2) always exploited in BDDC implementations
- Let us see how to exploit (3), in order to **reduce synchronization** and boost scalability (overlapped implementation)

Overlapped implementation

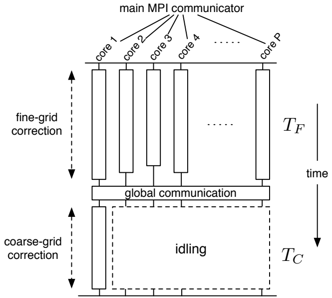
Typical parallel implementation



- All MPI tasks have f-g duties and one/several have also c-g duties
- Computation of f-g/c-g duties serialized (but they are independent!)
- $T_C \propto O(P^2) \rightarrow$ idling $\simeq PT_C$
- mem $\propto \mathcal{O}(P^{\frac{4}{3}}) \rightarrow$ mem per core rapidly exceeded

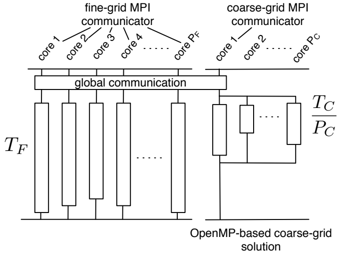
Overlapped implementation

Typical parallel implementation



- All MPI tasks have f-g duties and one/several have also c-g duties
- Computation of f-g/c-g duties serialized (but they are independent!)
- $T_C \propto O(P^2) \rightarrow$ idling $\simeq PT_C$
- mem $\propto \mathcal{O}(P^{\frac{4}{3}}) \rightarrow$ mem per core rapidly exceeded

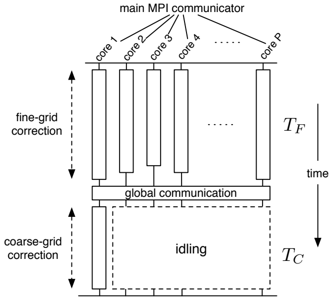
Highly-scalable parallel implementation



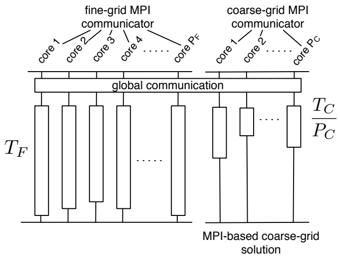
- MPI tasks have either f-g OR c-g duties
- f-g/c-g corrections OVERLAPPED in time (asynchronous)
- c-g tasks can be MASKED with f-g tasks duties
- MPI-based or OpenMP-based (this work) solutions are possible for c-g correction

Overlapped implementation

Typical parallel implementation



Highly-scalable parallel implementation



- All MPI tasks have f-g duties and one/several have also c-g duties
- Computation of f-g/c-g duties serialized (but they are independent!)
- $T_C \propto O(P^2) \rightarrow$ idling $\simeq PT_C$
- mem $\propto \mathcal{O}(P^{\frac{4}{3}}) \rightarrow$ mem per core rapidly exceeded

- MPI tasks have either f-g OR c-g duties
- f-g/c-g corrections OVERLAPPED in time (asynchronous)
- c-g tasks can be MASKED with f-g tasks duties
- MPI-based or OpenMP-based (this work) solutions are possible for c-g correction

Overlapping regions

Solve $Ax = b$ w/ BDDC-PCG

Precond'er set-up (M_{BDDC})
call PCG(A,M_{BDDC},b,x_0)

PCG

$r_0 := b - Ax_0$
$z_0 := M_{\mathrm{BDDC}}^{-1} r_0$
$p_0 := z_0$
for $j = 0, \ldots,$ till CONV **do**
$s_{j+1} = Ap_j$
$\ldots$
$z_{j+1} := M_{\mathrm{BDDC}}^{-1} r_{j+1}$
$\ldots$
end for

Overlapping regions

Fine-grid tasks	Coarse-grid task
Identify local coarse DoFs	
Construct G_{A_C} (Global comm'on)	
Symb fact($G_{A_F^{(i)}}$) $\mathcal{O}(n_i^{\frac{4}{3}})$ Symb fact($G_{A_{II}^{(i)}}$) $\mathcal{O}(n_i^{\frac{4}{3}})$ Num fact($A_F^{(i)}$) $\mathcal{O}(n_i^2)$ Compute Φ_i $\mathcal{O}(n_i^{\frac{4}{3}})$ $A_C^{(i)} := \Phi_i^t A^{(i)} \Phi_i$	Symb fact(G_{A_C}) $\mathcal{O}(P^{\frac{4}{3}})$
Gather $A_C^{(i)}$ (Global comm'on)	
Num fact($A_{II}^{(i)}$) $\mathcal{O}(n_i^2)$ $x_0 := x_0 - A_{II}^{-1} r_0$ $\mathcal{O}(n_i^{\frac{4}{3}})$ $r_0 := b - Ax_0$ $\mathcal{O}(n_i^{\frac{4}{3}})$ $r^{(i)} := I_i^t r$ $r_C^{(i)} := \Phi_i^t r^{(i)}$	$A_C :=$ assble($A_C^{(i)}$) Num fact(A_C) $\mathcal{O}(P^2)$
Gather $r_C^{(i)}$ (Global comm'on)	
Compute $s_F^{(i)}$ $\mathcal{O}(n_i^{\frac{4}{3}})$	$r_C :=$ assble($r_C^{(i)}$) Solve $A_C z_C = r_C$ $\mathcal{O}(P^{\frac{4}{3}})$
Scatter z_C into $z_C^{(i)}$ (Global comm'on)	
$s_C^{(i)} := \Phi_i z_C^{(i)}$	
$z^{(i)} := I_i(s_F^{(i)} + s_C^{(i)})$	

- Classify fine/coarse duties
- Map duties to f/c columns (+ synchro.)
- 3 overlapping regions (!)
- ALL coarse duties can be masked (!)

FEMPAR Software

FEMPAR (in-house developed HPC software, free software GNU-GPL):
Finite Element Multiphysics PARallel software

- Massively parallel sw for FE simulation of multiphysics PDEs
- Scalable preconditioning of fully coupled and implicit system via block preconditioning techniques (Part II)
- Scalable preconditioning for one-physics (elliptic) PDEs relies on BDDC
  → hybrid MPI/OpenMP implementation
- Relies on highly-efficient vendor implementations of the dense/sparse BLAS (Intel MKL, IBM ESSL, etc.), and interfaces to external multi-threaded sparse direct solvers (PARDISO, HSL_MA87, etc.) and serial AMG preconditioners (HSL_MI20)
- Free-software initiative funded by the ERC via a Proof of Concept Grant 2014

FEMPAR Software

FEMPAR (in-house developed HPC software, free software GNU-GPL):
Finite Element Multiphysics PARallel software

- Massively parallel sw for FE simulation of multiphysics PDEs
- Scalable preconditioning of fully coupled and implicit system via block preconditioning techniques (Part II)
- Scalable preconditioning for one-physics (elliptic) PDEs relies on BDDC
  → hybrid MPI/OpenMP implementation
- Relies on highly-efficient vendor implementations of the dense/sparse BLAS (Intel MKL, IBM ESSL, etc.), and interfaces to external multi-threaded sparse direct solvers (PARDISO, `HSL_MA87`, etc.) and serial AMG preconditioners (`HSL_MI20`)
- Free-software initiative funded by the ERC via a Proof of Concept Grant 2014

FEMPAR Software

FEMPAR (in-house developed HPC software, free software GNU-GPL):
Finite Element Multiphysics PARallel software

- Massively parallel sw for FE simulation of multiphysics PDEs
- Scalable preconditioning of fully coupled and implicit system via block preconditioning techniques (Part II)
- Scalable preconditioning for one-physics (elliptic) PDEs relies on BDDC
  $\rightarrow$ hybrid MPI/OpenMP implementation
- Relies on highly-efficient vendor implementations of the dense/sparse BLAS (Intel MKL, IBM ESSL, etc.), and interfaces to external multi-threaded sparse direct solvers (PARDISO, `HSL_MA87`, etc.) and serial AMG preconditioners (`HSL_MI20`)
- Free-software initiative funded by the ERC via a Proof of Concept Grant 2014

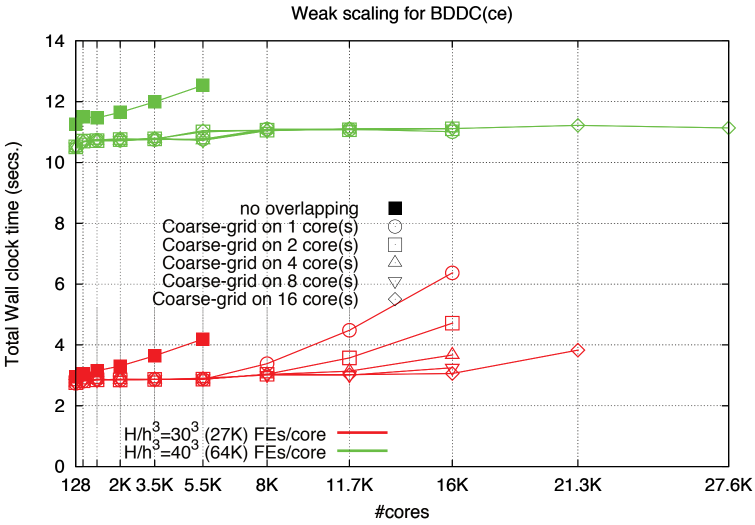
Weak scalability for 3D Poisson

Target machine: HELIOS@IFERC-CSC
4,410 bullx B510 compute blades (2 Intel Xeon E5-2680 8-core CPUs; 64GB)

- Target problem: $-\Delta u = f$ on $\overline{\Omega} = [0,2] \times [0,1] \times [0,1]$
- Uniform global mesh (Q1 FEs) + Uniform partition (cubic local meshes)
- $8, 432, \ldots, 27648$ cores for fine duties
- Direct solution of Dirichlet/Neumann/coarse problems (PARDISO)
- Entire 16-core blade for coarse-grid duties (multi-threaded PARDISO)
- Gradually larger local problem sizes: $\frac{H}{h} = 30^3, 40^3$ FEs/core

Weak scaling BDDC(corners+edges)

BDDC(corners+edges) :: Poisson problem



Outline

Why BDDC for extreme scales?

1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can be solved **inexactly** (AMG-cycle)
5. A **multilevel** extension is possible (for extreme core counts)

- (1)-(2)-(3) already exploited in our overlapped BDDC implementations
- Let us see how to exploit (4), in order to boost scalability further (overlapped/inexact implementation)

Inexact BDDC

- Exact (using direct solvers) BDDC is a very effective preconditioner
- But also a computationally/memory demanding one
- To reduce both demands, solve approximately internal problems (e.g., AMG)
- Numerical analysis: inexact BDDC also algorithmically scalable [Dohrmann, 2007]
- Benefit has to be viewed in light of future parallel architectures: the most scalable architectures (e.g., IBM BG) will have more limited memory per core
- Further, the coarse solver time increases as P instead of P^2, much less degradation for high core counts (due to linear complexity)

Inexact BDDC

- Exact (using direct solvers) BDDC is a very effective preconditioner
- But also a computationally/memory demanding one
- To reduce both demands, solve approximately internal problems (e.g., AMG)
- Numerical analysis: inexact BDDC also algorithmically scalable [Dohrmann, 2007]
- Benefit has to be viewed in light of future parallel architectures: the most scalable architectures (e.g., IBM BG) will have more limited memory per core
- Further, the coarse solver time increases as P instead of P^2, much less degradation for high core counts (due to linear complexity)

Inexact BDDC

- Exact (using direct solvers) BDDC is a very effective preconditioner
- But also a computationally/memory demanding one
- To reduce both demands, solve approximately internal problems (e.g., AMG)
- Numerical analysis: inexact BDDC also algorithmically scalable [Dohrmann, 2007]
- Benefit has to be viewed in light of future parallel architectures: the most scalable architectures (e.g., IBM BG) will have more limited memory per core
- Further, the coarse solver time increases as P instead of P^2, much less degradation for high core counts (due to linear complexity)

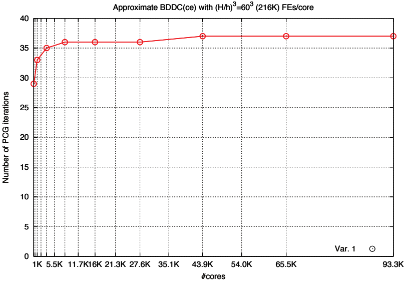
Weak scalability for 3D Poisson

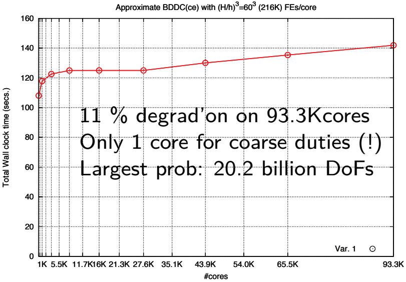
Target machine: JUQUEEN@JSC
28,672 compute nodes (16-core, 64-way threaded IBM PPC A2; 16 GB)

- Target problem: $-\Delta u = f$ on $\overline{\Omega} = [0,2] \times [0,1] \times [0,1]$
- Uniform global mesh (Q1 FEs) + Uniform partition (cubic local meshes)
- $8, 432, \ldots, 93312$ cores for fine duties
- Serial AMG preconditioners (`HSL_MI20`)
- 1 core for coarse-grid duties
- Fixed local problem size $\frac{H}{h} = 60^3$ FEs/core

Weak scaling inexact BDDC(c+e)

Inexact BDDC(corners+edges) :: Poisson problem, $\frac{H}{h} = 60$ (216K FEs/core)





# of outer solver iterations — Total time (secs.)

Outer solver	Φ	Dirichlet	Neumann	Coarse
PCG	AMG(2)	AMG(1)	AMG(2)	AMG(1)

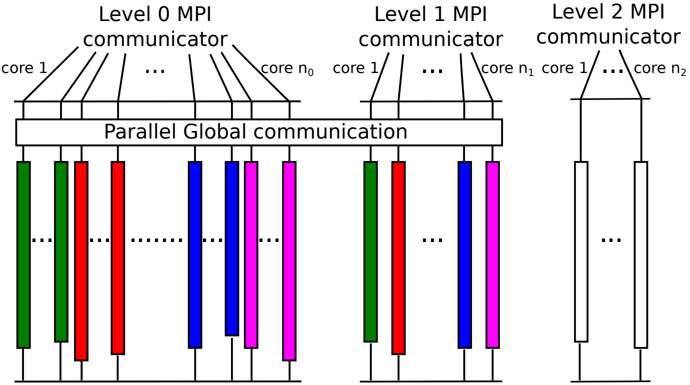
Outline

Why BDDC for extreme scales?

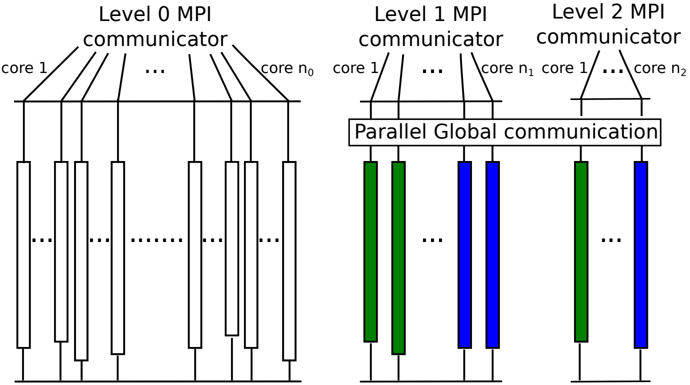
1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can be solved **inexactly** (AMG-cycle)
5. A **multilevel** extension is possible (for extreme core counts)

- (1)-(2)-(3)-(4) already exploited in our BDDC implementations
- Let us see how to exploit (5), in order to **go to extreme scales** (overlapped/inexact/multilevel implementation)

Overlapped multilevel



Overlapped multilevel

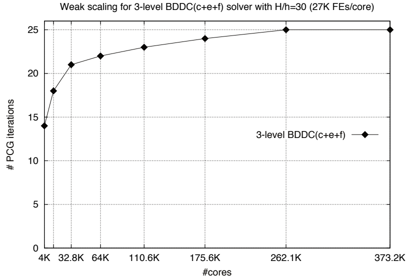


Weak scalability for 3D Poisson

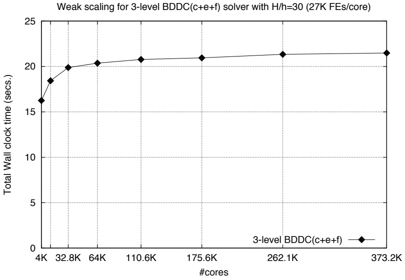
Target machine: JUQUEEN@JSC
28,672 compute nodes (16-core, 64-way threaded IBM PPC A2; 16 GB)

- Fixed local problem size $\frac{H}{h} = 30^3$ FEs/core
- 512 level-0 cores per level-1 cores
- 3 levels: e.g. $373,978 = 373,248(L0) + 729(L1) + 1(L2)$ cores
- Direct solution of Dirichlet/Neumann/coarse problems (PARDISO)
- DIRECT solvers (PARDISO)
- Results from yesterday!... GOD SAVE JUQUEEN!

Weak scaling 3-level BDDC(c+e+f) solver



# of PCG iters.



Total time (secs.)

Experiment set-up									
Lev.	# MPI tasks								FEs/core
1st	4K	13.8K	32.7K	64K	110.6K	175.6K	262.1K	373.2K	30^3 (27K)
2nd	8	27	64	125	216	343	512	729	8^3 (512)
3rd	1	1	1	1	1	1	1	1	n/a

Part II

Multiphysics solvers

Part II: Multiphysics solvers

HPSC
High Performance Scientific Computing

Outline

CIMNE

5 Motivation: CFD solvers

6 Recursive-block preconditioning

The Navier-Stokes system

The continuous problem:

$$\begin{aligned} \partial_t \mathbf{u} - \nu \Delta \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p &= \mathbf{f}, \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned}$$

The discrete problem (e.g. using Galerkin/stabilized FEM):

$$\begin{bmatrix} F & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ g \end{bmatrix},$$

Block preconditioning

1. Consider an exact block LU factorization:

$$\begin{bmatrix} F & G \\ D & C \end{bmatrix} = \begin{bmatrix} I & 0 \\ DF^{-1} & I \end{bmatrix} \begin{bmatrix} F & G \\ 0 & S \end{bmatrix}, \quad S = C - DF^{-1}G$$

2. Define an *inexact* block factorization, e.g.,

$$\begin{bmatrix} M_F & G \\ 0 & M_S \end{bmatrix}, \qquad M_F/M_S \text{ are preconditioners of } F/S$$

3. **Key ingredient**: scalable/robust M_F/M_S preconditioner, e.g., using BDDC

Solve $Mz = r$

1: Solve $M_S z_p = -r_p$
2: Solve $M_F z_u = r_u - B^T z_p$

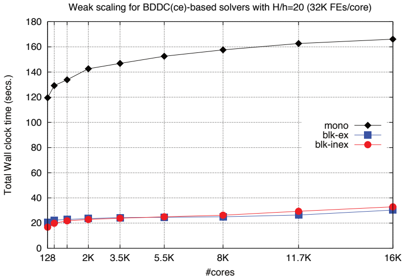
Weak scalability for 3D Stokes

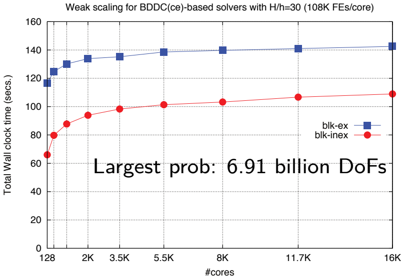
Target machine: HELIOS@IFERC-CSC
4,410 bullx B510 compute blades (2 Intel Xeon E5-2680 8-core CPUs)

- Target problem: Stokes on $\overline{\Omega} = [0,1]^3$ (lid-driven cavity problem)
- Uniform global mesh (Q1-Q1 FEs, ASGS-stabilized) + Uniform partition
- $8, 432, \ldots, 16000$ cores for fine duties
- Entire 16-core blade for coarse duties
- Three different preconditioners: (a) `mono`, (b) `blk-ex`, (c) `blk-inex`
- Direct (a & b) or AMG(1) (c) solves for Dirichlet/Neumann/Coarse probs.
- Gradually larger local problem sizes $\frac{H}{h} = 20, 30$

Weak scaling BDDC-based solvers

Stokes problem



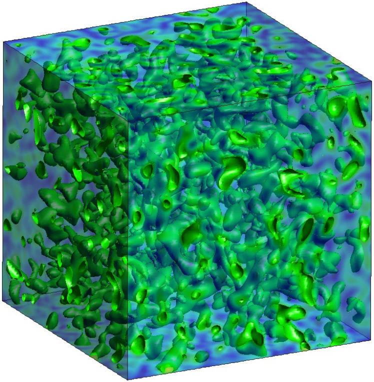


BDDC(c+e), 32K nodes/core   BDDC(c+e), 108K nodes/core

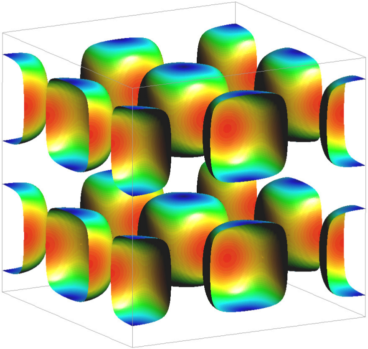
Total time (secs.)

Turbulent flows modelling

Applied to LES of turbulent incompressible flows



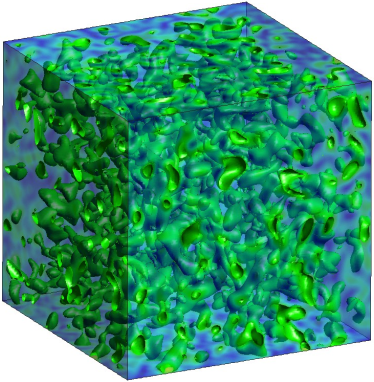
Decay Homogeneous isotropic turbulence (vorticity)



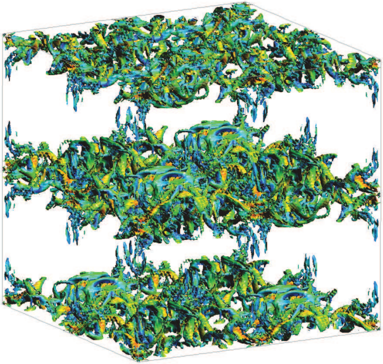
Taylor-Green vortex flow (vorticity $t = 1, 6$)

Turbulent flows modelling

Applied to LES of turbulent incompressible flows



Decay Homogeneous isotropic turbulence (vorticity)



Taylor-Green vortex flow (vorticity $t = 1, 6$)

HPSC
High Performance Scientific Computing

Outline

CIMNE

Recursive block preconditioning

Target: Scalable preconditioners for multiphysics (block precond + BDDC)

1. Reblock a generic multiphysics problem as a 2×2 block system:

$$\begin{bmatrix} F & G \\ D & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

2. Define an *inexact* block factorization, e.g.,

$$\begin{bmatrix} M_F & G \\ 0 & M_S \end{bmatrix}^{-1}, \qquad M_F/M_S \text{ are preconditioners of } F/S$$

3. If M_F and/or M_S involve two or more variables
   $\rightarrow$ recursively approximate by an incomplete LU (goto 1)

Recursive block preconditioning

Target: Scalable preconditioners for multiphysics (block precond + BDDC)

1. Reblock a generic multiphysics problem as a 2×2 block system:

$$\begin{bmatrix} F & G \\ D & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

2. Define an *inexact* block factorization, e.g.,

$$\begin{bmatrix} M_F & G \\ 0 & M_S \end{bmatrix}^{-1}, \qquad M_F/M_S \text{ are preconditioners of } F/S$$

3. If M_F and/or M_S involve two or more variables
   $\rightarrow$ recursively approximate by an incomplete LU (goto 1)

Recursive block preconditioning

Target: Scalable preconditioners for multiphysics (block precond + BDDC)

1. Reblock a generic multiphysics problem as a 2×2 block system:

$$\begin{bmatrix} F & G \\ D & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

2. Define an *inexact* block factorization, e.g.,

$$\begin{bmatrix} M_F & G \\ 0 & M_S \end{bmatrix}^{-1}, \qquad M_F/M_S \text{ are preconditioners of } F/S$$

3. If M_F and/or M_S involve two or more variables
   $\rightarrow$ recursively approximate by an incomplete LU (`goto` 1)

Recursive block preconditioning

Target: Scalable preconditioners for multiphysics (block precond + BDDC)

1. Reblock a generic multiphysics problem as a 2×2 block system:

$$\begin{bmatrix} F & G \\ D & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

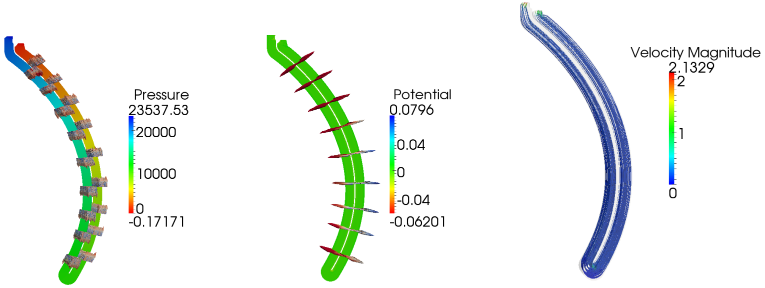
2. Define an *inexact* block factorization, e.g.,

$$\begin{bmatrix} M_F & G \\ 0 & M_S \end{bmatrix}^{-1}, \qquad M_F/M_S \text{ are preconditioners of } F/S$$

3. If M_F and/or M_S involve two or more variables
   $\rightarrow$ recursively approximate by an incomplete LU (`goto` 1)

Key ingredient: scalable and robust M_F and M_S preconditioner

Test blanket simulations



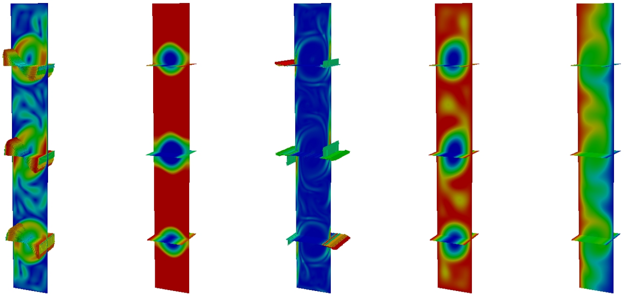
(a) u and p (b) j and ϕ (c) Velocity streamlines

Simulation results for the Tecnofus TBM.

- Typical blanket module simulation, $\mathrm{Ha} \simeq 15,000$

Test problem: 3D thermal cavity

Framework applied to thermal MHD (benchmarks)



(a) u (b) p (c) j (d) ϕ (e) T

Thermal cavity at Ha=100.

Abstract implementation of (block-)operators in FEMPAR [SB, Martín, Planas]

Conclusions and future work

- Highly scalable implementation of BDDC (overlapping)
- Inexact BDDC solvers highly scalable (hybrid DD-AMG)
- Multiscale BDDC implementations

- CFD solvers based on block preconditioners + BDDC
- Extension to multiphysics: recursive block-preconditioning
- Example of use: effective inductionless MHD

- Future work: Exploit our very recent ML extension and consider Multilevel/inexact/overlapped BDDC implementation

Conclusions and future work

- Highly scalable implementation of BDDC (overlapping)
- Inexact BDDC solvers highly scalable (hybrid DD-AMG)
- Multiscale BDDC implementations

- CFD solvers based on block preconditioners + BDDC
- Extension to multiphysics: recursive block-preconditioning
- Example of use: effective inductionless MHD

- Future work: Exploit our very recent ML extension and consider Multilevel/inexact/overlapped BDDC implementation

Conclusions and future work

- Highly scalable implementation of BDDC (overlapping)
- Inexact BDDC solvers highly scalable (hybrid DD-AMG)
- Multiscale BDDC implementations

- CFD solvers based on block preconditioners + BDDC
- Extension to multiphysics: recursive block-preconditioning
- Example of use: effective inductionless MHD

- Future work: Exploit our very recent ML extension and consider Multilevel/inexact/overlapped BDDC implementation

References

- S. Badia, A. F. Martín and J. Principe. A highly scalable parallel implementation of balancing domain decomposition by constraints. *SIAM Journal on Scientific Computing*. Vol. 36(2), pp. C190-C218, 2014.
- S. Badia, A. F. Martín and J. Principe. Implementation and scalability analysis of balancing domain decomposition methods. *Archives of Computational Methods in Engineering*. Vol. 20(3), pp. 239-262, 2013.
- S. Badia, A. F. Martín and J. Principe. On the scalability of inexact balancing domain decomposition by constraints with overlapped coarse/fine corrections. Submitted, 2014.
- S. Badia, A. F. Martín and R. Planas. Block recursive LU preconditioners for the thermally coupled incompressible inductionless MHD problem Journal of Computational Physics, Vol. 274, pp. 562-591, 2014.

Work funded by the ERC Starting Grant 258443 and Proof of Concept Grant

- Preprints at `http://badia.rmee.upc.edu/sbadia_ar.html`
- HPSC team: `https://web.cimne.upc.edu/groups/comfus/`

European Research Council
Established by the European Commission