A Highly Scalable Implementation of the BDDC Preconditioner

Alberto F. Martín, Santiago Badia, Javier Principe

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

Universitat Politècnica de Catalunya
Barcelona, Spain

SIAM Conference on Computational Science and Engineering
March 3, 2017
Atlanta, Georgia, USA

Outline

Outline

Problem statement

- **Variational problem**: Find $u \in H^1(\Omega)$ s.t.:

$$a(u, v) = (f, v), \qquad \forall v \in H_0^1(\Omega),$$

assuming $a(\cdot, \cdot)$ symmetric, coercive (e.g., Laplacian, Linear Elasticity)

- **Discrete problem**: Find $u_h \in V_h \subset H^1(\Omega)$ (conforming FE space) s.t.:

$$a(u_h, v_h) = (f, v_h), \qquad \forall v_h \in V_h^0.$$

- **Algebraic problem**: Find $x \in \mathbb{R}^n$ s.t.:

$$Ax = b,$$

where A is a large, sparse, SPD matrix

Problem statement

- **Variational problem**: Find $u \in H^1(\Omega)$ s.t.:
$$a(u, v) = (f, v), \qquad \forall v \in H^1_0(\Omega),$$
assuming $a(\cdot, \cdot)$ symmetric, coercive (e.g., Laplacian, Linear Elasticity)

- **Discrete problem**: Find $u_h \in V_h \subset H^1(\Omega)$ (conforming FE space) s.t.:
$$a(u_h, v_h) = (f, v_h), \qquad \forall v_h \in V_h^0.$$

- **Algebraic problem**: Find $x \in \mathbb{R}^n$ s.t.:
$$Ax = b,$$
where A is a large, sparse, SPD matrix

Problem statement

- **Variational problem**: Find $u \in H^1(\Omega)$ s.t.:

$$a(u, v) = (f, v), \qquad \forall v \in H_0^1(\Omega),$$

assuming $a(\cdot, \cdot)$ symmetric, coercive (e.g., Laplacian, Linear Elasticity)

- **Discrete problem**: Find $u_h \in V_h \subset H^1(\Omega)$ (conforming FE space) s.t.:

$$a(u_h, v_h) = (f, v_h), \qquad \forall v_h \in V_h^0.$$

- **Algebraic problem**: Find $x \in \mathbb{R}^n$ s.t.:

$$Ax = b,$$

where A is a large, sparse, SPD matrix

Problem statement

- **Variational problem**: Find $u \in H^1(\Omega)$ s.t.:

$$a(u, v) = (f, v), \qquad \forall v \in H_0^1(\Omega),$$

  assuming $a(\cdot, \cdot)$ symmetric, coercive (e.g., Laplacian, Linear Elasticity)

- **Discrete problem**: Find $u_h \in V_h \subset H^1(\Omega)$ (conforming FE space) s.t.:

$$a(u_h, v_h) = (f, v_h), \qquad \forall v_h \in V_h^0.$$

- **Algebraic problem**: Find $x \in \mathbb{R}^n$ s.t.:

$$Ax = b,$$

  where A is a large, sparse, SPD matrix

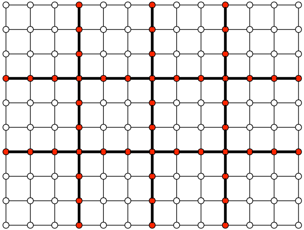
Motivation:

Efficient exploitation of current petascale supercomputers (and beyond)

Domain decomposition framework

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

V_h



Numerical solution of $Ax = b$

- Solve iteratively $M^{-1}Ax = M^{-1}b \rightarrow \tilde{A} = \tilde{b}$
- Preconditioner M aims at accelerating convergence
- M should be a parallel, cheap-to-invert, "good" approximation of A

Numerical solution of $Ax = b$

- Solve iteratively $M^{-1}Ax = M^{-1}b \rightarrow \tilde{A} = \tilde{b}$
- Preconditioner M aims at accelerating convergence
- M should be a parallel, cheap-to-invert, "good" approximation of A

Numerical solution of $Ax = b$

- Solve iteratively $M^{-1}Ax = M^{-1}b \rightarrow \tilde{A} = \tilde{b}$
- Preconditioner M aims at accelerating convergence
- M should be a parallel, cheap-to-invert, "good" approximation of A

Solve $Ax = b$ by M-PCG

Set-up M
call PCG(A,M,b,x^0)

Numerical solution of $Ax = b$

- Solve iteratively $M^{-1}Ax = M^{-1}b \rightarrow \tilde{A} = \tilde{b}$
- Preconditioner M aims at accelerating convergence
- M should be a parallel, cheap-to-invert, "good" approximation of A

Solve $Ax = b$ by M-PCG

Set-up M

call PCG(A,M,b,x^0)

PCG (In: (A, M, f, x^0), Out: x)

$r^0 := f - Ax^0$
$z^0 := \mathbf{M^{-1}} r^0$
$p^0 := z^0$
for $j = 0, \dots,$ till convergence **do**
  $s^{j+1} = Ap^j$
  $\alpha^j := (r^j, z^j)/(s^{j+1}, p^j)$
  $x^{j+1} := x^j + \alpha^j p^j$
  $r^{j+1} := r^j - \alpha^j s^j$
  $z^{j+1} := \mathbf{M^{-1}} r^{j+1}$
  $\beta^j := (r^{j+1}, z^{j+1})/(r^j, z^j)$
  $p^{j+1} := z^{j+1} + \beta^j p^j$
end for

Numerical solution of $Ax = b$

- Solve iteratively $M^{-1}Ax = M^{-1}b \to \tilde{A} = \tilde{b}$
- Preconditioner M aims at accelerating convergence
- M should be a parallel, cheap-to-invert, "good" approximation of A

Solve $Ax = b$ by M-PCG

Set-up M

call PCG(A,M,b,x^0)

PCG (In: (A, M, f, x^0), Out: x)

$r^0 := f - Ax^0$

$z^0 := \mathbf{M^{-1}} r^0$

$p^0 := z^0$

for $j = 0, \ldots,$ till convergence **do**

$s^{j+1} = Ap^j$

$\alpha^j := (r^j, z^j)/(s^{j+1}, p^j)$

$x^{j+1} := x^j + \alpha^j p^j$

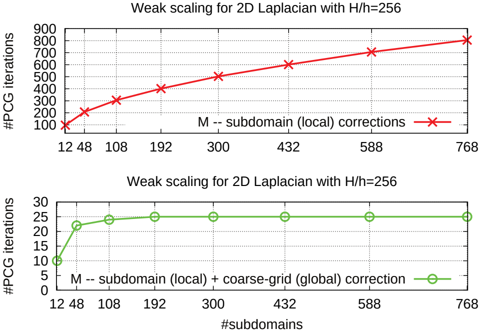
$r^{j+1} := r^j - \alpha^j s^j$

$z^{j+1} := \mathbf{M^{-1}} r^{j+1}$

$\beta^j := (r^{j+1}, z^{j+1})/(r^j, z^j)$

$p^{j+1} := z^{j+1} + \beta^j p^j$

end for



Weak scaling: $\uparrow P$, fixed $\frac{N}{P}$

Outline

BDDC Balancing DD by constraints

BDDC preconditioner [Dohrmann, 2003]

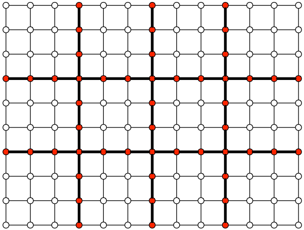
- Replace V_h by $\bar{V}_h$ (reduced continuity)
- Define the injection $E : \bar{V}_h \longrightarrow V_h$ (weighted average; involves nearest neighbours communication)
- Find $\bar{z}_h \in \bar{V}_h$ such that:

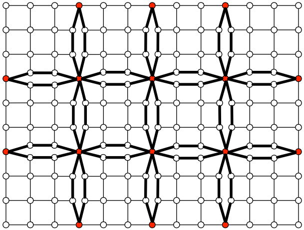
$$a(\bar{z}_h, \bar{v}_h) = (E^T r_h, \bar{v}_h), \quad \forall \bar{v}_h \in \bar{V}_h$$

and obtain $z_h = M_{BDDC}^{-1} r_h = \mathcal{E} E \bar{z}_h$

- Last correction $\mathcal{E}$ is the harmonic extension of the interface values (involves local Dirichlet solvers)

V_h





$\bar{V}_h$

BDDC Balancing DD by constraints

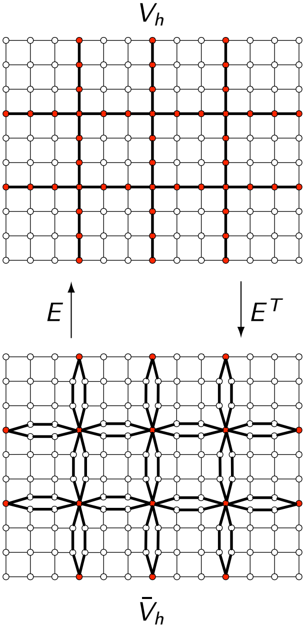
BDDC preconditioner [Dohrmann, 2003]

- Replace V_h by $\bar{V}_h$ (reduced continuity)
- Define the injection $E : \bar{V}_h \longrightarrow V_h$ (weighted average; involves nearest neighbours communication)
- Find $\bar{z}_h \in \bar{V}_h$ such that:

$$a(\bar{z}_h, \bar{v}_h) = (E^T r_h, \bar{v}_h), \quad \forall \bar{v}_h \in \bar{V}_h$$

and obtain $z_h = M_{BDDC}^{-1} r_h = \mathcal{E} E \bar{z}_h$

- Last correction $\mathcal{E}$ is the harmonic extension of the interface values (involves local Dirichlet solvers)



BDDC Balancing DD by constraints

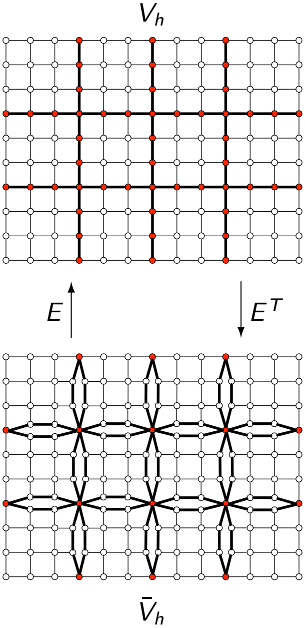
BDDC preconditioner [Dohrmann, 2003]

- Replace V_h by $\bar{V}_h$ (reduced continuity)
- Define the injection $E : \bar{V}_h \longrightarrow V_h$ (weighted average; involves nearest neighbours communication)
- Find $\bar{z}_h \in \bar{V}_h$ such that:

$$a(\bar{z}_h, \bar{v}_h) = (E^T r_h, \bar{v}_h), \quad \forall \bar{v}_h \in \bar{V}_h$$

and obtain $z_h = M_{BDDC}^{-1} r_h = \mathcal{E} E \bar{z}_h$

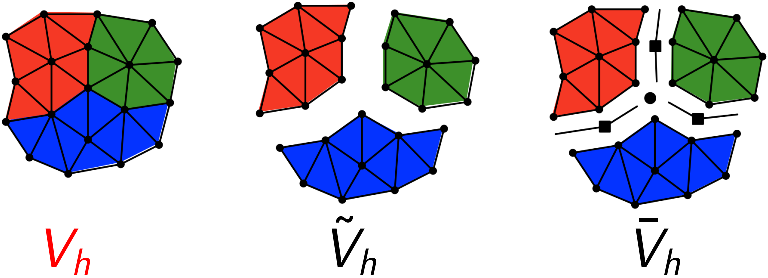
- Last correction $\mathcal{E}$ is the harmonic extension of the interface values (involves local Dirichlet solvers)



BDDC preconditioning

Given a conforming FE space V_h, we have:

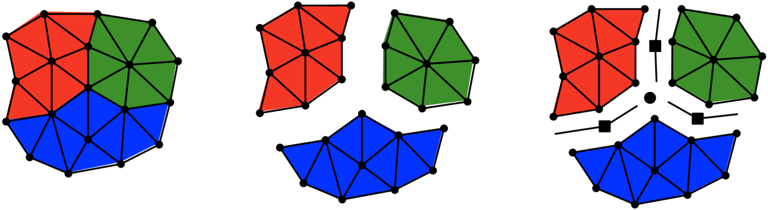
- The unassembled space $\tilde{V}_h = V_h^1 \times \ldots \times V_h^n$ $(V_h \subset \tilde{V}_h)$
- The BDDC (under-assembled) space $\bar{V}_h = \{\tilde{v}_h \in \tilde{V}_h | \text{ continuous } \mathcal{F}(\tilde{v}_h)\}$



BDDC preconditioning

Given a conforming FE space V_h, we have:

- The unassembled space $\tilde{V}_h = V_h^1 \times \ldots \times V_h^n$ $(V_h \subset \tilde{V}_h)$
- The BDDC (under-assembled) space $\bar{V}_h = \{\tilde{v}_h \in \tilde{V}_h |$ continuous $\mathcal{F}(\tilde{v}_h)\}$

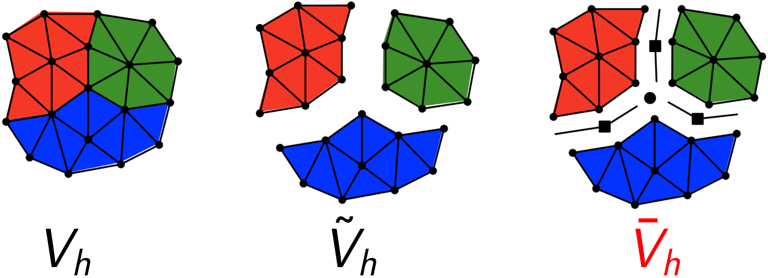


V_h $\tilde{V}_h$ $\bar{V}_h$

BDDC preconditioning

Given a conforming FE space V_h, we have:

- The unassembled space $\tilde{V}_h = V_h^1 \times \ldots \times V_h^n$ $(V_h \subset \tilde{V}_h)$
- The BDDC (under-assembled) space $\bar{V}_h = \{\tilde{v}_h \in \tilde{V}_h |$ continuous $\mathcal{F}(\tilde{v}_h)\}$

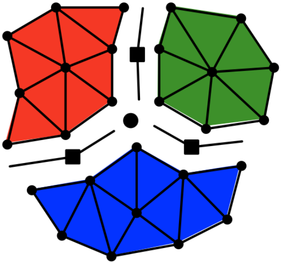


BDDC preconditioning

- The under-assembled space $\bar{V}_h$ can be decomposed as:

$$\bar{V}_h = \bar{V}_h^F \oplus \bar{V}_h^C, \text{ with} \begin{cases} \bar{V}_h^F = \{\bar{v}_h \in \bar{V}_h | \mathcal{F}(\bar{v}_h) = 0\} \\ \bar{V}_h^C = \{\bar{v}_h \in \bar{V}_h | \bar{v}_h \perp_{\tilde{\mathcal{A}}} \bar{V}_h^F\} \end{cases}$$

- Sought-after correction split into fine-grid ($\bar{z}_F$) + coarse-grid ($\bar{z}_C$) correction



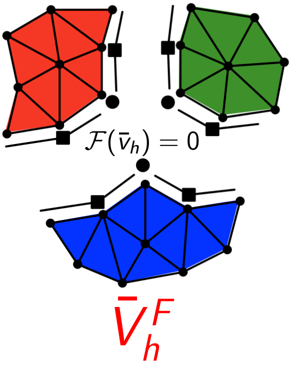
$\bar{V}_h$

BDDC preconditioning

- The under-assembled space $\bar{V}_h$ can be decomposed as:

$$\bar{V}_h = \bar{V}_h^F \oplus \bar{V}_h^C, \text{ with} \begin{cases} \bar{V}_h^F = \{\bar{v}_h \in \bar{V}_h | \mathcal{F}(\bar{v}_h) = 0\} \\ \bar{V}_h^C = \{\bar{v}_h \in \bar{V}_h | \bar{v}_h \perp_{\tilde{\mathcal{A}}} \bar{V}_h^F\} \end{cases}$$

- Sought-after correction split into fine-grid ($\bar{z}_F$) + coarse-grid ($\bar{z}_C$) correction



- The under-assembled space $\bar{V}_h$ can be decomposed as:

$$\bar{V}_h = \bar{V}_h^F \oplus \bar{V}_h^C, \text{ with} \begin{cases} \bar{V}_h^F = \{\bar{v}_h \in \bar{V}_h | \mathcal{F}(\bar{v}_h) = 0\} \\ \bar{V}_h^C = \{\bar{v}_h \in \bar{V}_h | \bar{v}_h \perp_{\tilde{\mathcal{A}}} \bar{V}_h^F\} \end{cases}$$

- Sought-after correction split into fine-grid $(\bar{z}_F)$ + coarse-grid $(\bar{z}_C)$ correction

Fine-grid correction $(\bar{z}_F)$

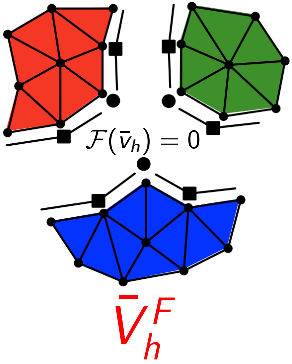
- Find $\bar{z}_F \in \mathbb{R}^{\tilde{n}}$ such that

$$\begin{bmatrix} \tilde{A} & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \bar{z}_F \\ \lambda \end{bmatrix} = \begin{bmatrix} E^T r \\ 0 \end{bmatrix}$$

- Equivalent to P independent problems

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

$$\begin{bmatrix} \tilde{A}^{(i)} & C^{(i)T} \\ C^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \bar{z}_F^{(i)} \\ \lambda^{(i)} \end{bmatrix} = \begin{bmatrix} E^{(i)T} r^{(i)} \\ 0 \end{bmatrix}$$

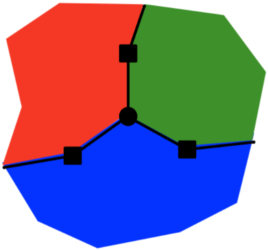


BDDC preconditioning

- The under-assembled space $\bar{V}_h$ can be decomposed as:

$$\bar{V}_h = \bar{V}_h^F \oplus \bar{V}_h^C, \text{ with} \begin{cases} \bar{V}_h^F = \{\bar{v}_h \in \bar{V}_h | \mathcal{F}(\bar{v}_h) = 0\} \\ \bar{V}_h^C = \{\bar{v}_h \in \bar{V}_h | \bar{v}_h \perp_{\tilde{\mathcal{A}}} \bar{V}_h^F\} \end{cases}$$

- Sought-after correction split into fine-grid ($\bar{z}_F$) + coarse-grid ($\bar{z}_C$) correction



$\bar{V}_h^C$

BDDC preconditioning

- The under-assembled space $\bar{V}_h$ can be decomposed as:

$$\bar{V}_h = \bar{V}_h^F \oplus \bar{V}_h^C, \text{ with } \begin{cases} \bar{V}_h^F = \{\bar{v}_h \in \bar{V}_h | \mathcal{F}(\bar{v}_h) = 0\} \\ \bar{V}_h^C = \{\bar{v}_h \in \bar{V}_h | \bar{v}_h \perp_{\tilde{A}} \bar{V}_h^F\} \end{cases}$$

- Sought-after correction split into fine-grid ($\bar{z}_F$) + coarse-grid ($\bar{z}_C$) correction

Coarse-grid correction ($\bar{z}_C$)

Computation of $\bar{V}_h^C = \mathrm{span}\{\Phi_1, \Phi_2, \ldots, \Phi_{n_C}\}$

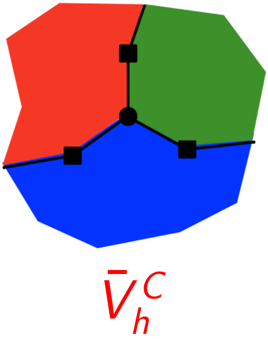
- Find $\Phi \in \mathbb{R}^{\tilde{n} \times n_C}$ such that

$$\begin{bmatrix} \tilde{A} & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Phi \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ I_{n_C} \end{bmatrix}$$

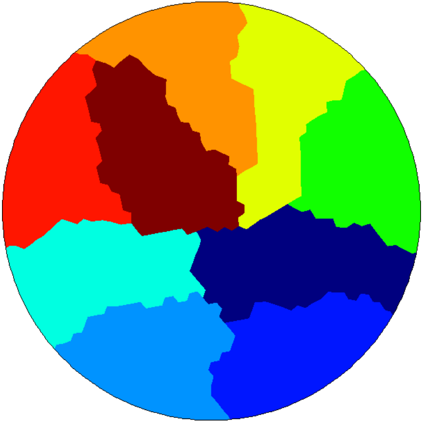
- Equivalent to P independent problems

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

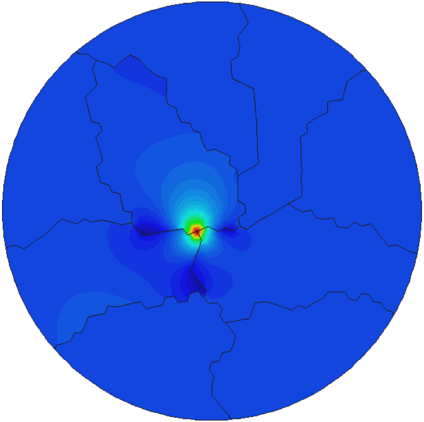
$$\begin{bmatrix} \tilde{A}^{(i)} & C^{(i)T} \\ C^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \Phi^{(i)} \\ \Lambda^{(i)} \end{bmatrix} = \begin{bmatrix} 0 \\ I_{n_C^{(i)}} \end{bmatrix}$$



BDDC coarse space basis functions

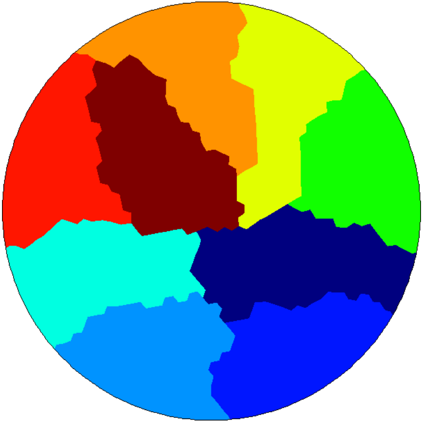


Circle domain partitioned into 9 subdomains

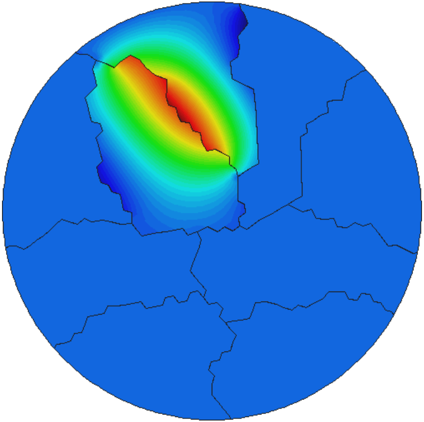


Φ_j ($\bar{V}_h^C$'s corner basis vector)

BDDC coarse space basis functions



Circle domain partitioned into 9 subdomains



Φ_j ($\bar{V}_h^C$'s edge basis vector)

BDDC preconditioning

- The under-assembled space $\bar{V}_h$ can be decomposed as:

$$\bar{V}_h = \bar{V}_h^F \oplus \bar{V}_h^C, \text{ with } \begin{cases} \bar{V}_h^F = \{\bar{v}_h \in \bar{V}_h | \mathcal{F}(\bar{v}_h) = 0\} \\ \bar{V}_h^C = \{\bar{v}_h \in \bar{V}_h | \bar{v}_h \perp_{\tilde{A}} \bar{V}_h^F\} \end{cases}$$

- Sought-after correction split into fine-grid ($\bar{z}_F$) + coarse-grid ($\bar{z}_C$) correction

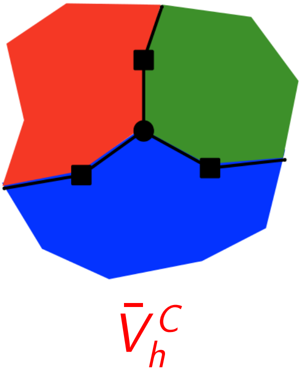
Coarse-grid correction ($\bar{z}_C$)

Assembly and solution of coarse-grid problem:

- $A_C = \text{assembly}(\Phi^{(i)T} \tilde{A}^{(i)} \Phi^{(i)})$
- $r_C = \text{assembly}(\Phi^{(i)T} E^{(i)T} r^{(i)})$
- Solve $A_C \alpha_C = r_C$
- $\bar{z}_C = \Phi \alpha_C$

Coarse-grid problem is:

- Global, i.e. couples all subdomains
- But much smaller than original problem
- Potential loss of parallel efficiency with P



BDDC preconditioning

- The under-assembled space $\bar{V}_h$ can be decomposed as:

$$\bar{V}_h = \bar{V}_h^F \oplus \bar{V}_h^C, \text{ with} \begin{cases} \bar{V}_h^F = \{\bar{v}_h \in \bar{V}_h | \mathcal{F}(\bar{v}_h) = 0\} \\ \bar{V}_h^C = \{\bar{v}_h \in \bar{V}_h | \bar{v}_h \perp_{\tilde{A}} \bar{V}_h^F\} \end{cases}$$

- Sought-after correction split into fine-grid ($\bar{z}_F$) + coarse-grid ($\bar{z}_C$) correction

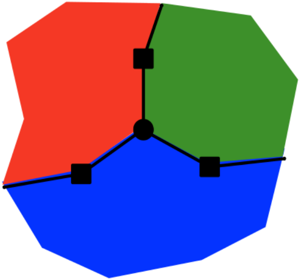
Coarse-grid correction ($\bar{z}_C$)

Assembly and solution of coarse-grid problem:

- $A_C = \text{assembly}(\Phi^{(i)T}\tilde{A}^{(i)}\Phi^{(i)})$
- $r_C = \text{assembly}(\Phi^{(i)T}E^{(i)T}r^{(i)})$
- Solve $A_C\alpha_C = r_C$
- $\bar{z}_C = \Phi\alpha_C$

Coarse-grid problem is:

- Global, i.e. couples all subdomains
- But much smaller than original problem
- Potential loss of parallel efficiency with P



$\bar{V}_h^C$

Coarse DoFs definition

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

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

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)

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?

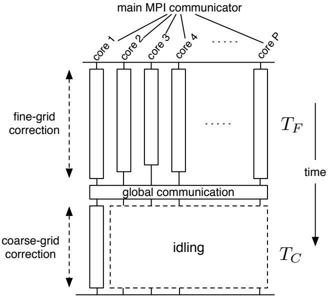
1. (Mathematically supported) **extremely aggressive coarsening** ($10^5 - 10^6$ size reduction between fine/coarse level)
2. The coarse matrix has a similar **sparsity** as the original matrix
3. Coarse/local components can be computed **in parallel** (like additive)
4. ALL local + coarse problems can 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)

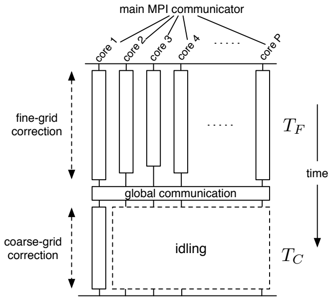
- (1)-(2) always exploited in BDDC implementations
- Let us see how to exploit (3), in order to introduce **asynchronicity** and boost scalability (overlapped implementation)

Naive parallel implementation



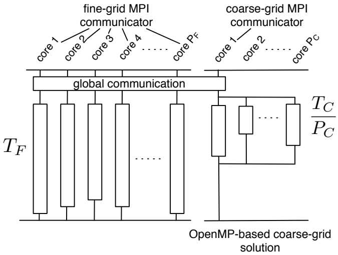
- All MPI tasks have f-g duties and one/several have also c-g duties
- Computation of f-g/c-g duties serialized (but they are independent!)
- $T_C \propto O(P^2) \rightarrow$ idling $\simeq PT_C$
- mem $\propto \mathcal{O}(P^{\frac{4}{3}}) \rightarrow$ mem per core rapidly exceeded

Naive 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
Our approach



- MPI tasks have either f-g OR c-g duties
- f-g/c-g corrections OVERLAPPED in time
- c-g duties can be MASKED with f-g ones duties
- OpenMP/MPI-based (MPI later on this talk) solutions possible for c-g correction

Overlapping regions

Solve $Ax = b$ by BDDC-PCG

Set-up M_{BDDC}
$r^0 := b - Ax^0$
$x^0 := x^0 + R_I A_{II}^{-1} R_I^t r^0$
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

Solve $Ax = b$ by BDDC-PCG

Set-up M_{BDDC}
$r^0 := b - Ax^0$
$x^0 := x^0 + R_I A_{II}^{-1} R_I^t r^0$
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

Fine-grid tasks	Coarse-grid task
Identify local coarse DoFs	
Gather coarse-grid DoFs	
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 $A_C^{(i)}$	
Num fact($A_{II}^{(i)}$) $\mathcal{O}(n_i^2)$ $x_0 := x_0 + R_I A_{II}^{-1} R_I^t r_0$ $\mathcal{O}(n_i^{\frac{4}{3}})$ $r_0 := b - Ax_0$ $r^{(i)} := E_i^t r$ $r_C^{(i)} := \Phi_i^t r^{(i)}$	$A_C :=$ assble($A_C^{(i)}$) Num fact(A_C) $\mathcal{O}(P^2)$
Gather $r_C^{(i)}$	
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 $z_C^{(i)}$	
$s_{\mathrm{C}}^{(i)} := \Phi_i z_C^{(i)}$ $z^{(i)} := E_i(s_F^{(i)} + s_{\mathrm{C}}^{(i)})$ $z_I^{(i)} := -(A_{II}^{(i)})^{-1} A_{I\Gamma}^{(i)} z_\Gamma^{(i)}$ $\mathcal{O}(n_i^{\frac{4}{3}})$	

Weak scalability for 3D Laplacian

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

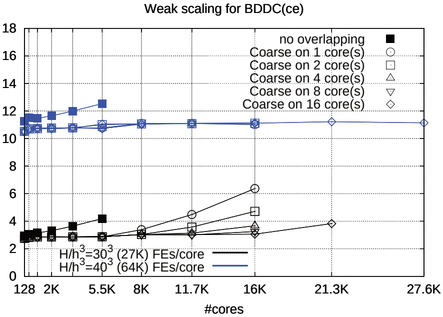
- Target problem: $-\Delta u = f$ on $\overline{\Omega} = [0,2] \times [0,1] \times [0,1]$
- Uniform global mesh (Q1 FEs) + Uniform partition (cubic local meshes)
- $8, 432, \ldots, 27648$ cores for fine duties
- Direct solution of Dirichlet/Neumann/coarse problems (PARDISO)
- Entire 16-core blade for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

Weak scaling 2-level BDDC

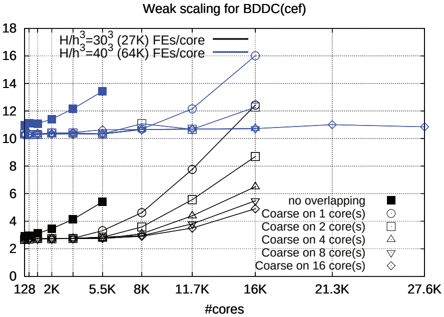
3D Laplacian problem on HELIOS@IFERC

4,410 bullx B510 compute blades (2 Intel Xeon E5-2680 8-core CPUs; 64GB)

Largest problem size is 1.8 billion DoFs



BDDC(ce)



BDDC(cef)

Total time (secs.)

Outline

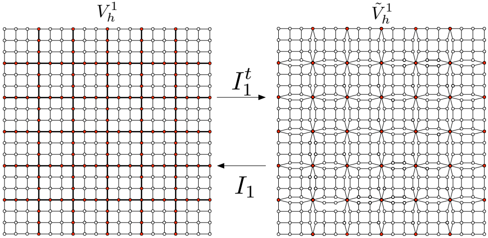
Why BDDC for extreme scales?

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 codes
- Let us see how to exploit (5), in order to boost scalability even further (for exact version for the moment)

MLBDDC basic idea

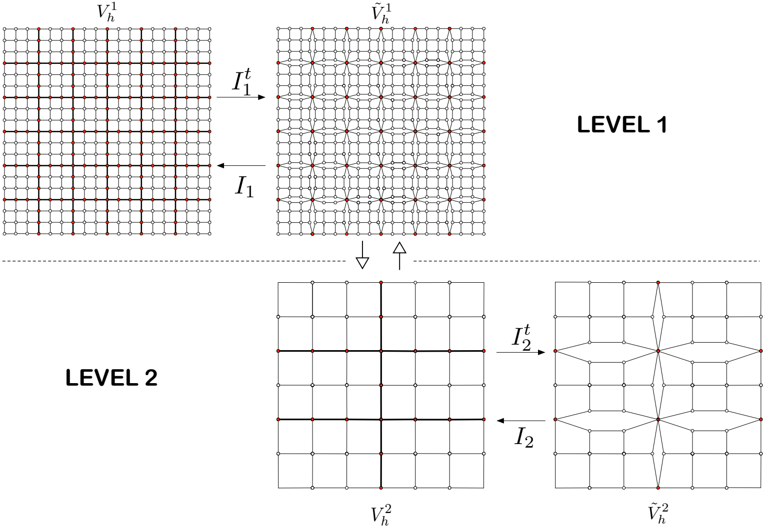
MLBDDC [Mandel et. al., 2008]: Replace coarse problem by BDDC precond



LEVEL 1

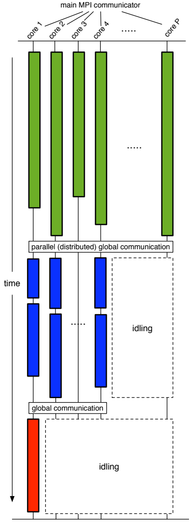
MLBDDC basic idea

MLBDDC [Mandel et. al., 2008]: Replace coarse problem by BDDC precond

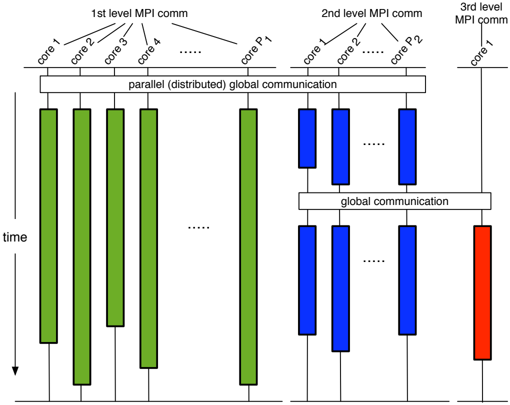


Highly scalable implementation of MLBDDC

Naive parallel implementation



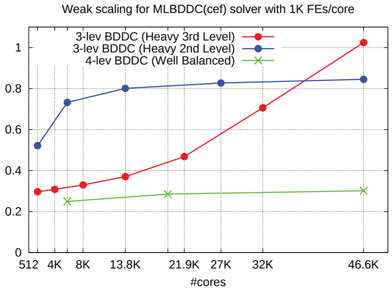
Overlapped implementation
Our approach



Goal: strike a balance such that blue/red areas are kept below green ones!

L_1 MPI tasks	L_2 MPI tasks	L_3 MPI task
Identify local coarse DoFs		
Gather coarse-grid DoFs		
Algorithm 1 ($k \equiv i_{L_1}$) Algorithm 2 ($k \equiv i_{L_1}$)	Build $G_{A_C^{(j_{L_2})}}$ Identify local coarse DoFs	
Compute $\Phi_{i_{L_1}}$	Gather coarse-grid DoFs	
$A_C^{(i_{L_1})} \leftarrow \Phi^t_{i_{L_1}}(-C^T_{i_{L_1}}\Lambda_{i_{L_1}})$	Algorithm 1 ($k \equiv i_{L_2}$)	Build G_{A_C}
Gather $A_C^{(i_{L_1})}$		Re+Sy fact(G_{A_C})
Algorithm 3 ($k \equiv i_{L_1}$) Algorithm 4 ($k \equiv i_{L_1}$)	$A_C^{(j_{L_2})} := \text{assemb}(A_C^{(i_{L_1})})$ Algorithm 2 ($k \equiv i_{L_2}$) Compute $\Phi_{i_{L_2}}$ $A_C^{(i_{L_2})} \leftarrow \Phi^t_{i_{L_2}}(-C^T_{i_{L_2}}\Lambda_{i_{L_2}})$	
	Gather $A_C^{(i_{L_2})}$	
	Algorithm 3 ($k \equiv i_{L_2}$)	$A_C := \text{assemb}(A_C^{(i_{L_2})})$ Num fact(A_C)
Gather $r_C^{(i_{L_1})}$		
Algorithm 5 ($k \equiv i_{L_1}$)	$r_C^{(j_{L_2})} := \text{assemb}(r_C^{(i_{L_1})})$ Algorithm 4 ($k \equiv i_{L_2}$)	
	Gather $r_C^{(i_{L_2})}$	
	Algorithm 5 ($k \equiv i_{L_2}$)	$r_C := \text{assemb}(r_C^{(i_{L_2})})$ Solve $A_C z_C = r_C$
	Scatter z_C into $z_C^{(i_{L_2})}$	
	Algorithm 6 ($k \equiv i_{L_2}$)	
Scatter $z_C^{(j_{L_2})}$ into $z_C^{(i_{L_1})}$		
Algorithm 6 ($k \equiv i_{L_1}$)		

Weak scaling (1K FEs/core)



Total Time (secs.)

Algorithm 1

Re+Sy fact($G_{A_F^{(k)}}$)

Re+Sy fact($G_{A_{II}^{(k)}}$)

Algorithm 2

Num fact($A_F^{(k)}$)

Algorithm 3

Num fact($A_{II}^{(k)}$)

Algorithm 4

$\delta_I^{(k)} \leftarrow (A_{II}^{(k)})^{-1} r_I^{(k)}$

$r_\Gamma^{(k)} \leftarrow r_\Gamma^{(k)} - A_{\Gamma I}^{(k)} \delta_I^{(k)}$

$r^{(k)} \leftarrow E_k^t r$

Algorithm 5

Solve

$$A_F^{(k)} \begin{bmatrix} t \\ s_F^{(k)} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ r^{(k)} \\ 0 \end{bmatrix}$$

Algorithm 6

$s_C^{(k)} \leftarrow \Phi_k z_C^{(k)}$

$z^{(k)} \leftarrow E_k(s_F^{(k)} + s_C^{(k)})$

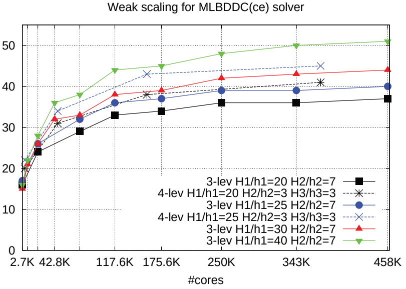
$z_I^{(k)} \leftarrow -(A_{II}^{(k)})^{-1} A_{I\Gamma}^{(k)} z_\Gamma^{(k)}$

$z_I^{(k)} \leftarrow z_I^{(k)} + \delta_I^{(k)}$

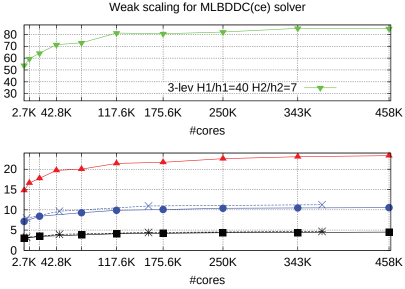
Weak scaling 3-lev BDDC(ce) solver

3D Laplacian problem on IBM BG/Q (JUQUEEN@JSC)
16 MPI tasks/compute node, 1 OpenMP thread/MPI task

Largest problem size is 29.2 billion DoFs



#PCG iterations



Total time (secs.)

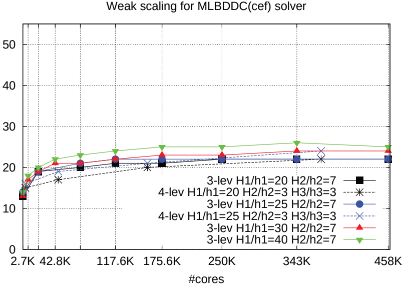
Experiment set-up

Lev.	# MPI tasks							FEs/core
1st	42.8K	74.1K	117.6K	175.6K	250K	343K	456.5K	$20^3/25^3/30^3/40^3$
2nd	125	216	343	512	729	1000	1331	7^3
3rd	1	1	1	1	1	1	1	n/a

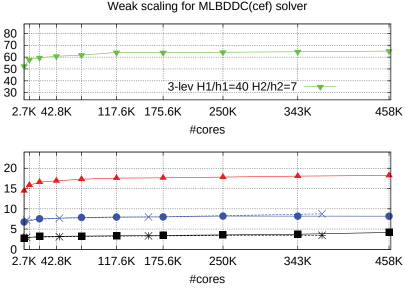
Weak scaling 3-lev BDDC(cef) solver

3D Laplacian problem on IBM BG/Q (JUQUEEN@JSC)
16 MPI tasks/compute node, 1 OpenMP thread/MPI task

Largest problem size is 29.2 billion DoFs



#PCG iterations



Total time (secs.)

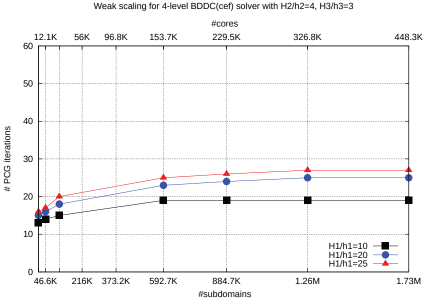
Experiment set-up								
Lev.	# MPI tasks							FEs/core
1st	42.8K	74.1K	117.6K	175.6K	250K	343K	456.5K	$20^3/25^3/30^3/40^3$
2nd	125	216	343	512	729	1000	1331	7^3
3rd	1	1	1	1	1	1	1	n/a

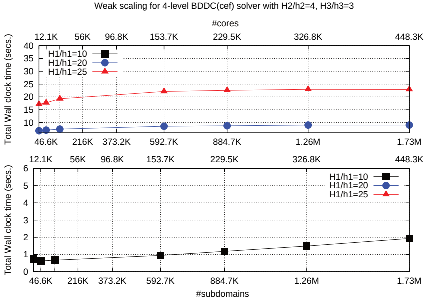
Weak scal. 4-lev BDDC(cef)+4 MPI tasks/core

3D Laplacian problem on IBM BG/Q (JUQUEEN@JSC)

64 MPI tasks/compute node, 1 OpenMP thread/MPI task

Largest problem size is 27 billion DoFs





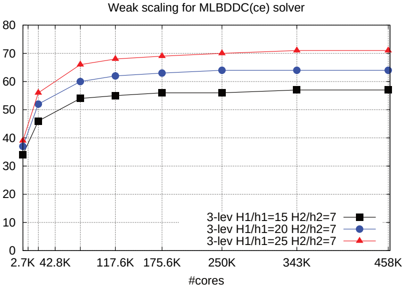
PCG iters. Total time (secs.)

Lev.	# MPI tasks							FEs/core
1st	110.6K	216K	373.2K	592.7K	884.7K	1.26M	1.73M	**10^3/20^3/25^3**
2nd	1.73K	3.38K	5.83K	9.26K	13.8K	19.7K	64K	4^3
3rd	64	125	216	343	512	729	1K	3^3
4th	1	1	1	1	1	1	1	n/a

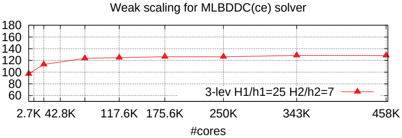
Weak scaling 3-lev BDDC(ce) solver

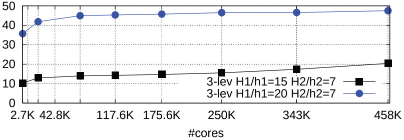
3D Linear Elasticity problem on IBM BG/Q (JUQUEEN@JSC)
16 MPI tasks/compute node, 1 OpenMP thread/MPI task

Largest problem size is 21.4 billion DoFs



#PCG iterations



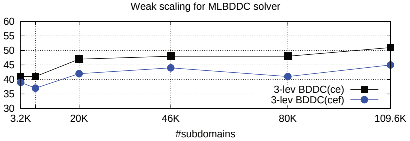


Total time (secs.)

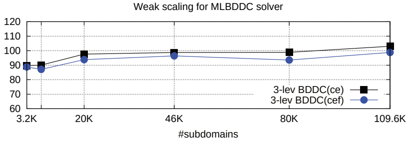
Experiment set-up								
Lev.	# MPI tasks							FEs/core
1st	42.8K	74.1K	117.6K	175.6K	250K	343K	456.5K	**15^3/20^3/25^3**
2nd	125	216	343	512	729	1000	1331	7^3
3rd	1	1	1	1	1	1	1	n/a

Weak scaling 3-lev BDDC

3D Laplacian problem + **unstructured mesh discretizations**
16 MPI tasks/compute node, 1 OpenMP thread/MPI task



#PCG iterations



Total time (secs.)

- Unstructured meshes of tetrahedra
- Each L_2 subdomain aggregates ≈ 384 L_1 subdomains
- Largest unstructured problem has ~ 3.6 billion FEs
- Remarkable scalability despite underlying irregularity of the problem

Outline

Conclusions

- Highly scalable implementation of (linear/exact) MLBDDC
  - Fully-distributed
  - Communicator-aware
  - Interlevel-overlapped (coarse-grain comput./comput./comm. overlap)
  - Recursive (extensible to arbitrary # levels)
- Remarkable scalability
  - 3D Laplacian and Linear Elasticity PDEs
  - 3/4 levels are sufficient to (efficiently) scale till full JUQUEEN
  - Largest scaling/problem sizes reported so far with (linear/exact) DDM

Conclusions

- Highly scalable implementation of (linear/exact) MLBDDC
  - Fully-distributed
  - Communicator-aware
  - Interlevel-overlapped (coarse-grain comput./comput./comm. overlap)
  - Recursive (extensible to arbitrary # levels)
- Remarkable scalability
  - 3D Laplacian and Linear Elasticity PDEs
  - 3/4 levels are sufficient to (efficiently) scale till full JUQUEEN
  - Largest scaling/problem sizes reported so far with (linear/exact) DDM

FEMPAR

- The algorithms presented herein have been implemented in FEMPAR:
  - OO framework for the development of parallel multiphysics FE solvers
  - From desktops/laptops to extreme scale supercomputers
  - More than >200K Fortran2008 code long
  - JSC High-Q club member since 2014
  - Open source project (GNU GPLv3)
- At the solver's level, FEMPAR provides an advanced OO user-customizable framework for the development of highly scalable MLBDDC precond's:
  - Tight integration among discretization/preconditioner (not black-box)
  - Up-to-know: "standard" MLBDDC [this talk], (r)PB-BDDC [Badia, Martín, Nguyen, Submitted'16], BDD by constraints+perturbation [Badia, Nguyen, SISC'16], Inexact BDDC [Badia, Martín, Principe, ParCo'15]
  - On-going: (ML)BDDC for unfitted FEM [Badia's talk SIAM CSE17], Inexact MLBDDC (hybrid DD/AMG)
  - Future plans: MLBDDC suitable for $\mathcal{H}(\mathbf{curl})$ and $\mathcal{H}(\mathbf{div})$ FE spaces
  - Other extensions that we envision possible are: deluxe scaling, spectral methods ... (volunteers?)

https://gitlab.com/fempar/fempar

Thank you!

Santiago Badia, Alberto F. Martín and Hieu Nguyen Physics-based balancing domain decomposition by constraints for heterogeneous problems. Submitted, 2016.

Santiago Badia, A. F. Martín and J. Principe. Multilevel balancing domain decomposition at extreme scales. *SIAM Journal on Scientific Computing*. Vol. 38(1), pp. C22–C52, 2016.

Santiago 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.