Scalable Domain Decomposition Preconditioners
For Heterogeneous Elliptic Problems

Pierre Jolivet, F. Hecht,
F. Nataf, C. Prud’homme

Laboratoire Jacques-Louis Lions
Laboratoire Jean Kuntzmann
INRIA Rocquencourt

10th Workshop of the INRIA-Illinois-ANL Joint Laboratory

November 26th, 2013
Consider the linear system: \( Au = f \in \mathbb{R}^n \).
A short introduction to DDM

Consider the linear system: \( Au = f \in \mathbb{R}^n \).

Given a decomposition of \([1; n]\), \((\mathcal{N}_1, \mathcal{N}_2)\), define:

- the restriction operator \( R_i \) from \([1; n]\) into \( \mathcal{N}_i \),
- \( R_i^T \) as the extension by 0 from \( \mathcal{N}_i \) into \([1; n]\).
A short introduction to DDM

Consider the linear system: \( Au = f \in \mathbb{R}^n \).

Given a decomposition of \([1; n]\), \((\mathcal{N}_1, \mathcal{N}_2)\), define:

- the restriction operator \( R_i \) from \([1; n]\) into \( \mathcal{N}_i \),
- \( R_i^T \) as the extension by 0 from \( \mathcal{N}_i \) into \([1; n]\).

Then solve concurrently:

\[
\begin{align*}
    u_1^{m+1} &= u_1^m + A_{11}^{-1} R_1(f - A_{1}^m) \\
    u_2^{m+1} &= u_2^m + A_{22}^{-1} R_2(f - A_{2}^m)
\end{align*}
\]

where \( u_i = R_i u \) and \( A_{ij} := R_i A R_j^T \).
A short introduction II

We have effectively divided, but we have yet to conquer.

_Duplicated_ unknowns coupled via a _partition of unity_:

\[
l = \sum_{i=1}^{N} R_i^T D_i R_i.
\]
A short introduction II

We have effectively divided, but we have yet to conquer.

*Duplicated* unknowns coupled via a *partition of unity*:

\[
I = \sum_{i=1}^{N} R_i^T D_i R_i.
\]

Then,

\[
\begin{align*}
  u^{m+1} &= \sum_{i=1}^{N} R_i^T D_i u_i^{m+1}.
\end{align*}
\]
A short introduction II

We have effectively divided, but we have yet to conquer.

_Duplicated_ unknowns coupled via a _partition of unity_:

$$I = \sum_{i=1}^{N} R_i^T D_i R_i.$$  

Then,  

$$u^{m+1} = \sum_{i=1}^{N} R_i^T D_i u_i^{m+1}.$$  

$$M^{-1} = \sum_{i=1}^{N} R_i^T D_i A_{ii}^{-1} R_i.$$
Contributions and goals

Based on algebraic results with the p. of u., we propose:

1. a reformulation of the global matrix-vector product eliminating the need of a global ordering,

2. a construction of a so-called “coarse operator” to enhance a simple preconditioner.

We are interested in the solution of various SPD systems, independently of:

- the discretization order,
- the contrast in the coefficients,
- the number of subdomains.
Using the overlap to its fullest extent

DDM methods are seldom used as standalone solvers.

Krylov methods and overlapping Schwarz methods

\[ Au \implies \text{efficient global matrix-vector product} \]
Using the overlap to its fullest extent

DDM methods are seldom used as standalone solvers.

Krylov methods and overlapping Schwarz methods

\[ Au = \sum_{j=1}^{N} AR_j^T D_j R_j u \]
Using the overlap to its fullest extent

DDM methods are seldom used as standalone solvers.

Krylov methods and overlapping Schwarz methods

\[ R_iAu = \sum_{j=1}^{N} R_iAR_j^T D_jR_ju \]
Using the overlap to its fullest extent

DDM methods are seldom used as standalone solvers.

Krylov methods and overlapping Schwarz methods

\[
R_i Au = \sum_{j=1}^{N} R_i A R_j^T D_j R_j u = \sum_{j \in \mathcal{O}_i} A_{ij} D_j R_j u
\]

\(\mathcal{O}_i\) are the neighbors of \(\Omega_i\), \(\overline{\mathcal{O}_i} = \mathcal{O}_i \cup \{i\}\).
Using the overlap to its fullest extent

DDM methods are seldom used as standalone solvers.

Krylov methods and overlapping Schwarz methods

\[ R_i A u = \sum_{j=1}^{N} R_i A R_j^T D_j R_j u = \sum_{j \in \mathcal{O}_i} A_{ij} D_j R_j u \]

\[ = \sum_{j \in \mathcal{O}_i} R_i R_j^T A_{jj} D_j R_j u \]

- no need for the global matrix, only local to neighbors mappings. ← explicit point-to-point communications via \( R_i R_j^T \).
- reuse of the operators from the preconditioner, \( A_{ii} \).

\( \mathcal{O}_i \) are the neighbors of \( \Omega_i \), \( \overline{\mathcal{O}_i} = \mathcal{O}_i \cup \{i\} \).
Limitations of one-level methods

One-level methods don’t require exchange of global information.

This hampers numerical scalability of such preconditioners.
A common technique in the field of DDM, MG, deflation: introduce an auxiliary "coarse" problem.

Let $Z$ be a rectangular matrix. Define

$$E := Z^T A Z.$$  

$Z$ has $O(N)$ columns, hence $E$ is much smaller than $A$. 
Two-level preconditioners I

A common technique in the field of DDM, MG, deflation: introduce an auxiliary “coarse” problem.

Let $Z$ be a rectangular matrix. Define

$$E := Z^T A Z.$$  

$Z$ has $\mathcal{O}(N)$ columns, hence $E$ is much smaller than $A$. Enrich the original preconditioner, e.g. additively

$$P^{-1} = M^{-1} + ZE^{-1}Z^T,$$

c.f. (Tang et al. 2009).
Two-level preconditioners II

The construction of $Z$ and the assembly of $E$ are challenging.

Let each domain compute concurrently $\nu_i$ vectors $\{\Lambda_{ij}\}_{j=1}^{\nu_i}$.

Define local dense rectangular matrices:

$$W_i = \begin{bmatrix} D_i \Lambda_{i1} & D_i \Lambda_{i2} & \cdots & D_i \Lambda_{i\nu_i} \end{bmatrix}.$$ 

Then, define the global deflation matrix as:

$$Z = \begin{bmatrix} R_1^T W_1 & R_2^T W_2 & \cdots & R_N^T W_N \end{bmatrix}$$
Generalized eigenvalue problems

For theoretical justification of $Z$, see (Spillane et al. 2011). Solved by ARPACK concurrently:

$$A_i^N \Lambda_j = \lambda_j D_i R_{i,0}^T R_{i,0} A_i^N D_i \Lambda_j$$

where

- $A_i^N$ is the local unassembled matrix,
- $R_{i,0}$ is the restriction from $\Omega_i$ to $\Omega_i \cap \bigcap_{j \in \mathcal{O}_i} \Omega_j$. 
Workflow during one coarse operator correction

How to compute $ZE^{-1} Z^T u \in \mathbb{R}^n$?
Workflow during one coarse operator correction

How to compute $Z^T u \in \mathbb{R}^n$?

\[
Z^T u = \begin{bmatrix}
\text{operations & MPI\_Gather}
\end{bmatrix}
\times
= \begin{bmatrix}
m \ll n
\end{bmatrix}
\]
Workflow during one coarse operator correction

How to compute $E^{-1}Z^T u \in \mathbb{R}^n$?

$Z^T u = \begin{array}{c}
\text{operations \\ \\ & MPI\_Gather \\ \\ & + linear solve}
\end{array}$
Workflow during one coarse operator correction

How to compute $ZE^{-1}Z^Tu \in \mathbb{R}^n$?

$$Z^T u = \begin{bmatrix} \text{operations} \\ \text{MPI}_\text{Gather} \end{bmatrix} + \text{linear solve} + \begin{bmatrix} \text{MPI}_\text{Scatter} \\ \text{operations} \end{bmatrix} = Z(Z^T AZ)^{-1}Z^T u$$
Workflow during one coarse operator correction

How to compute $Z E^{-1} Z^T u \in \mathbb{R}^n$?

$$Z^T u = \begin{array}{c}
\begin{array}{c}
\text{operations} \\
\text{MPI\_Gather} \end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\text{linear solve} \\
\text{MPI\_Scatter} \end{array}
\end{array} \begin{array}{c}
\begin{array}{c}
\text{operations} \end{array}
\end{array} = Z (Z^T AZ)^{-1} Z^T u$$

Communication pattern $\implies$ global reduction at the coarse level.
Workflow during one coarse operator correction

How to compute $ZE^{-1}Z^T u \in \mathbb{R}^n$?

$Z^T u = \begin{pmatrix} n \end{pmatrix} \times m \ll n$

$(Z^T AZ)^{-1} Z^T u = Z(Z^T AZ)^{-1} Z^T u$

operations & MPI_Gather + linear solve + MPI_Scatter & operations

Communication pattern $\Rightarrow$ global reduction at the coarse level.
Distribution of the coarse operator

How can one solve $E^{-1}z = c \in \mathbb{R}^m$?

Some constraints:

1. $E$ cannot be centralized on a single MPI process,
2. $E$ cannot be distributed on all MPI processes,
3. the solution must be computed fast and reliably.
Distribution of the coarse operator

How can one solve $E^{-1}z = c \in \mathbb{R}^m$?

Some constraints:

1. $E$ cannot be centralized on a single MPI process,
2. $E$ cannot be distributed on all MPI processes,
3. the solution must be computed fast and reliably.

$\implies$ use a direct solver with a distributed matrix on few master processes (number chosen at runtime).
Recalling $E = ZAZ^T$, it can be proven that the block $(i,j)$

$$E_{ij} = W_i^T A_{ij} W_j$$
$$E_{ij} = W_i^T R_i R_j^T A_{jj} W_j$$

1. compute locally $T_i = A_{ii} W_i$ (csrmm),
2. send to each neighbor, $S_j = R_j R_i^T T_i$,
3. receive from each neighbor $U_j = R_i R_j^T T_j$,
4. compute locally $E_{i,i} = W_i^T T_i$ (gemm),
5. compute locally $E_{i,j} = W_i^T U_j$ (gemm).

Note:  
- steps 2 and 3 overlap with step 4,
- if $j \not\in \mathcal{O}_i$, $R_i R_j^T = 0$.  

Assembly for Schwarz methods
Example of heterogeneous coefficients

\[ \nabla \cdot (\kappa \nabla u) = f + \text{BC} \]
2D geometry

\[(E_1, E_2) = (200, 0.01) \text{ GPa}\]
\[(\nu_1, \nu_2) = (0.25, 0.45)\]

\[\nabla \cdot \sigma = f + \text{BC}\]
3D geometry
Machine used for scaling runs

Curie Thin Nodes

- 5,040 compute nodes.
- 2 eight-core Intel Sandy Bridge@2.7 GHz per node.
- IB QDR full fat tree.
- 1.7 PFLOPs peak performance.
Strong scaling (linear elasticity)

1 subdomain/MPI process, 2 OpenMP threads/MPI process.

Runtime (seconds)

--- Linear speedup
- - - - -

3D \((\mathbb{P}_2 \text { FE})\)

2D \((\mathbb{P}_3 \text { FE})\)
Strong scaling (linear elasticity)

1 subdomain/MPI process, 2 OpenMP threads/MPI process.

2.1B d.o.f. in 2D ($P_3$ FE)  300M d.o.f. in 3D ($P_2$ FE)
Weak scaling (scalar diffusion equation)

1 subdomain/MPI process, 2 OpenMP threads/MPI process.

Numerical results

Coarse space preconditioners

Conclusion
Weak scaling (scalar diffusion equation)

1 subdomain/MPI process, 2 OpenMP threads/MPI process.

2.1M $\frac{\text{d.o.f.}}{\text{sbdmn}}$ in 2D ($\mathbb{P}_4$ FE)

280k $\frac{\text{d.o.f.}}{\text{sbdmn}}$ in 3D ($\mathbb{P}_2$ FE)

Time (seconds)

#processes: 256, 512, 1024, 2048, 4096, 8192

- Factorization
- Deflation vectors
- Coarse operator
- Krylov method
Distributed global matrix

Local to global mapping \[\Rightarrow\] distribution of the global matrix à la PETSc (split row-wise).

Comparing performance of setup and solution phases between our solver against purely algebraic (+ near null space) solvers:

- GASM – one-level domain decomposition method (ANL),
- Hypre BoomerAMG – algebraic multigrid (LLNL),
- GAMG – algebraic multigrid (ANL/LBL).
Solution of a linear system I

Homogeneous 3D Poisson equation discretized by $P_1$ FE solved on 2,048 MPI processes, 111M d.o.f.
Solution of a linear system II

Heterogeneous 3D linear elasticity equation discretized by $\mathbb{P}_2$ FE solved on 2,048 MPI processes, 127M d.o.f.
Final words

Limitations:

• scaling of the coarse operator in 3D beyond 10k subdomains,
• deflation vectors need elementary matrices to be computed.

Summary:

• scalable framework for building two-level preconditioners for both Schwarz or substructuring methods (FETI-1),
• easily interfacable (FEM, FVM) without a global ordering.

Outlooks:

• adaptive (re)construction/recycling of the coarse operator,
• nonlinear and saddle point problems.
Final words

Limitations:

• scaling of the coarse operator in 3D beyond 10k subdomains,
• deflation vectors need elementary matrices to be computed.

Summary:

• scalable framework for building two-level preconditioners for both Schwarz or substructuring methods (FETI-1),
• easily interfacable (FEM, FVM) without a global ordering.

Outlooks:

• adaptive (re)construction/recycling of the coarse operator,
• nonlinear and saddle point problems.

Thank you !

Solvers parameters

- Schwarz GenEO: $\nu_i = 20$, overlap $= 1$ (geometric).
- PETSc GASM: overlap $= 10$ (algebraic).
- Hyper BoomerAMG: HMIS coarsening, extended “classical” interpolation, no CF-relaxation, 2 levels of aggressive coarsening.
- PETSc GAMG: 1 smoothing step, -mg_levels_ksp_type richardson -mg_levels_pc_type sor.

OpenMPI bindings for hybrid runs:

```
--bind-to-socket --bycore.
```
Distribution of $E$ when built with $N = 16$ using 4 masters. On the right, the number of values per master is roughly the same if the values below the diagonal are dropped (symmetric coarse operator).
## Timings for assembling the coarse operator

### 3D

| \(N\) | \(P\) | \(\text{dim}(E)\) | \(|\mathcal{O}_i|\) (average) | Memory cost of “\(E^{-1}\)” | Time |
|---|---|---|---|---|---|
| 256 | 4 | 5120 | 11.5 | 38 MB | 2.78 s |
| 512 | 6 | 10240 | 12.4 | 78 MB | 3.35 s |
| 1024 | 8 | 20480 | 12.0 | 156 MB | 93 MB | 4.42 s | 11.25 s |
| 2048 | 12 | 40960 | 12.9 | 332 MB | 138 MB | 6.91 s | 5.68 s |
| 4096 | 18 | 73728 | 13.7 | 434 MB | 172 MB | 10.75 s | 8.04 s |
| 8192 | 64 | 131072 | 14.6 | 420 MB | 241 MB | 19.92 s | 17.30 s |

### 2D

| \(N\) | \(P\) | \(\text{dim}(E)\) | \(|\mathcal{O}_i|\) (average) | Memory cost of “\(E^{-1}\)” | Time |
|---|---|---|---|---|---|
| 256 | 2 | 5376 | 5.5 | 21 MB | 9.39 s |
| 512 | 4 | 10240 | 5.6 | 32 MB | 9.96 s |
| 1024 | 8 | 20480 | 5.5 | 65 MB | 57 MB | 9.92 s | 10.14 s |
| 2048 | 12 | 38912 | 5.7 | 94 MB | 83 MB | 10.05 s | 6.20 s |
| 4096 | 18 | 81920 | 5.8 | 99 MB | 73 MB | 10.87 s | 5.10 s |
| 8192 | 36 | 163840 | 5.8 | 152 MB | 118 MB | 13.27 s | 6.96 s |
### Strong scaling (linear elasticity)

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>Factorization</th>
<th>Deflation</th>
<th>Solution</th>
<th>#it.</th>
<th>Total</th>
<th>#d.o.f.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D</td>
<td>1024</td>
<td>177.86 s</td>
<td>264.03 s</td>
<td>77.41 s</td>
<td>28</td>
<td>530.56 s</td>
<td>$293.98 \cdot 10^6$</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>62.69 s</td>
<td>97.29 s</td>
<td>20.39 s</td>
<td>23</td>
<td>186.04 s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>19.64 s</td>
<td>35.70 s</td>
<td>9.73 s</td>
<td>20</td>
<td>73.12 s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8192</td>
<td>6.33 s</td>
<td>22.08 s</td>
<td>6.05 s</td>
<td>27</td>
<td>51.76 s</td>
<td></td>
</tr>
<tr>
<td>2D</td>
<td>1024</td>
<td>37.01 s</td>
<td>131.76 s</td>
<td>34.29 s</td>
<td>28</td>
<td>213.20 s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>17.55 s</td>
<td>53.83 s</td>
<td>17.52 s</td>
<td>28</td>
<td>95.10 s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>6.90 s</td>
<td>27.07 s</td>
<td>8.64 s</td>
<td>23</td>
<td>47.71 s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8192</td>
<td>2.01 s</td>
<td>20.78 s</td>
<td>4.79 s</td>
<td>23</td>
<td>34.54 s</td>
<td></td>
</tr>
</tbody>
</table>
Weak scaling (scalar diffusion equation)

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>Factorization</th>
<th>Deflation</th>
<th>Solution</th>
<th>#it.</th>
<th>Total</th>
<th>#d.o.f.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D</td>
<td>256</td>
<td>64.24 s</td>
<td>117.74 s</td>
<td>15.81 s</td>
<td>13</td>
<td>200.57 s</td>
<td>74.62 \cdot 10^6</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>63.97 s</td>
<td>112.17 s</td>
<td>19.93 s</td>
<td>18</td>
<td>199.41 s</td>
<td>144.70 \cdot 10^6</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>63.22 s</td>
<td>118.58 s</td>
<td>16.18 s</td>
<td>14</td>
<td>202.40 s</td>
<td>288.80 \cdot 10^6</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>59.43 s</td>
<td>117.59 s</td>
<td>21.34 s</td>
<td>17</td>
<td>205.26 s</td>
<td>578.01 \cdot 10^6</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>58.14 s</td>
<td>110.68 s</td>
<td>27.89 s</td>
<td>20</td>
<td>207.47 s</td>
<td>1.15 \cdot 10^9</td>
</tr>
<tr>
<td></td>
<td>8192</td>
<td>54.96 s</td>
<td>116.64 s</td>
<td>23.64 s</td>
<td>17</td>
<td>215.15 s</td>
<td>2.31 \cdot 10^9</td>
</tr>
<tr>
<td>2D</td>
<td>256</td>
<td>29.40 s</td>
<td>111.35 s</td>
<td>25.71 s</td>
<td>29</td>
<td>175.85 s</td>
<td>695.96 \cdot 10^6</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>29.60 s</td>
<td>111.52 s</td>
<td>27.99 s</td>
<td>28</td>
<td>179.07 s</td>
<td>1.39 \cdot 10^9</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>29.43 s</td>
<td>112.18 s</td>
<td>33.63 s</td>
<td>28</td>
<td>185.16 s</td>
<td>2.79 \cdot 10^9</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>29.18 s</td>
<td>112.23 s</td>
<td>33.74 s</td>
<td>28</td>
<td>185.20 s</td>
<td>5.58 \cdot 10^9</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>29.80 s</td>
<td>113.69 s</td>
<td>31.02 s</td>
<td>26</td>
<td>185.38 s</td>
<td>11.19 \cdot 10^9</td>
</tr>
<tr>
<td></td>
<td>8192</td>
<td>29.83 s</td>
<td>113.81 s</td>
<td>30.67 s</td>
<td>25</td>
<td>187.57 s</td>
<td>22.31 \cdot 10^9</td>
</tr>
</tbody>
</table>