MUMPS ON THOUSANDS OF CORES: FEEDBACK ON THE USE OF DIRECT SOLVERS IN DDM

Pierre Jolivet, IRIT-CNRS

MUMPS User Days
June 2

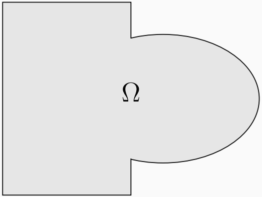
INTRODUCTION

- open-source, `https://github.org/hpddm/hpddm`
- handles (optimized) Schwarz or substructuring methods,
- provides adaptive coarse space constructions,
- interfaced with MUMPS, PaStiX, PARDISO, Dissection, SuiteSparse, BoomerAMG,
- interfaced with LAPACK and ARPACK,
- can be used with FreeFem++, Feel++, or as is in C, C++, Python or Fortran.

Solving Laplace's equation with Feel++

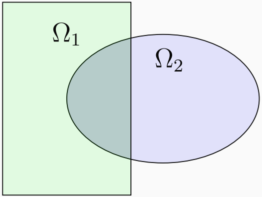
```
auto mesh = loadMesh(_mesh = new Mesh<Simplex<2>>);
auto Vh = Pch<2>(mesh);
auto u = Vh->element(), v = Vh->element();
auto f = expr("2*x*y+cos(y):x:y");
// a(u,v) = ∫_Ω ∇u · ∇v
auto a = form2(_trial = Vh, _test = Vh);
a = integrate(_range = elements(mesh),
        _expr = gradt(u) * trans(grad(v)));
// l(v) = ∫_Ω fv
auto l = form1(_test = Vh);
l = integrate(_range = elements(mesh),
        _expr = f * id(v));
// u = 0 on ∂Ω
a += on(_range = boundaryfaces(mesh),
        _rhs = l, _element = u,
        _expr = cst(0.0));
a.solve(_rhs = l, _solution = u);
```

Consider the linear system: $Au = f \in \mathbb{K}^n$.



Consider the linear system: $Au = f \in \mathbb{K}^n$.
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 $[\![1; n]\!]$.

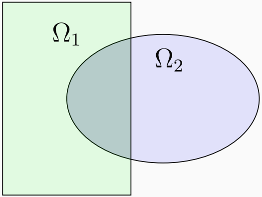


Consider the linear system: $Au = f \in \mathbb{K}^n$.
Given a decomposition of $[\![1;n]\!]$, $(\mathcal{N}_1, \mathcal{N}_2)$, define:

- 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 $[\![1;n]\!]$.

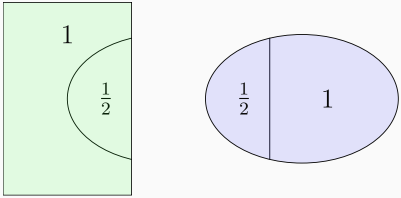
Then define:

$$u_i = R_i u \qquad A_{ij} = R_i A R_j^T.$$



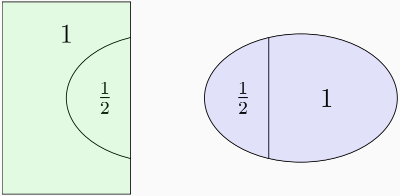
Duplicated unknowns coupled via a *partition of unity*:

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



Duplicated unknowns coupled via a *partition of unity*:

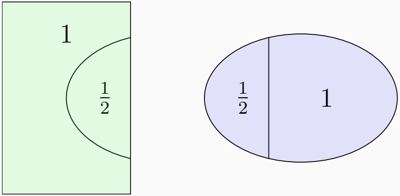
$$I = \sum_{i=1}^{N} R_i^T D_i R_i.$$



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

Duplicated unknowns coupled via a *partition of unity*:

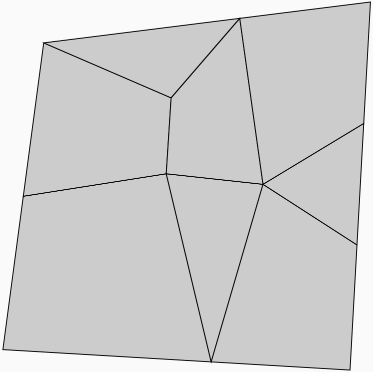
$$I = \sum_{i=1}^{N} R_i^T D_i R_i.$$



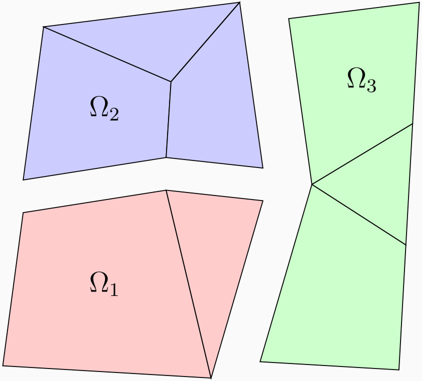
Then, $u^{m+1} = \sum_{i=1}^{N} R_i^T D_i u_i^{m+1}$.

$$M_{\text{RAS}}^{-1} = \sum_{i=1}^{N} R_i^T D_i A_{ii}^{-1} R_i$$

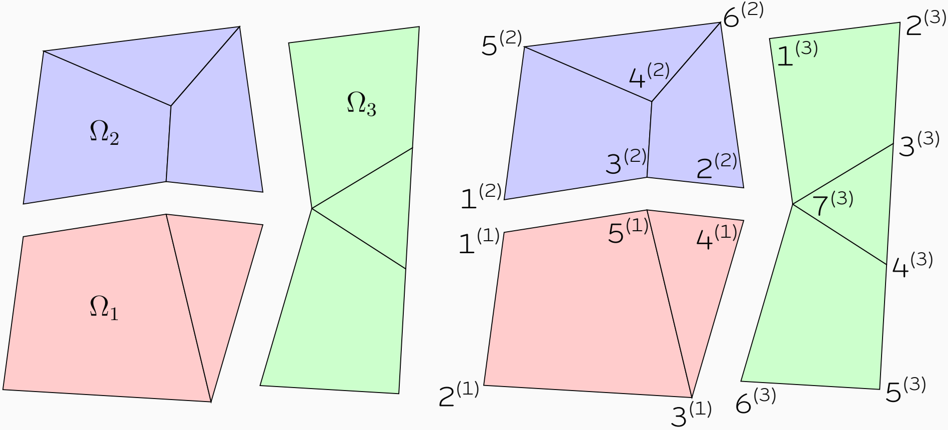
[Cai and Sarkis 1999]



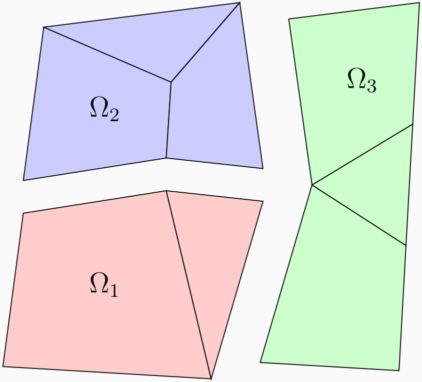
[Gosselet and Rey 2006]



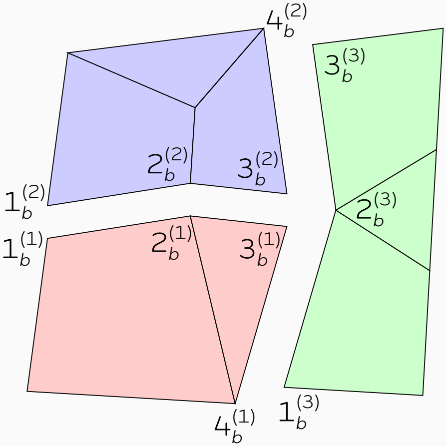
Subdomain tearing



[Gosselet and Rey 2006]   Local numbering $A^{(k)} = \begin{bmatrix} A_{ii} & A_{ib} \\ A_{bi} & A_{bb} \end{bmatrix}$

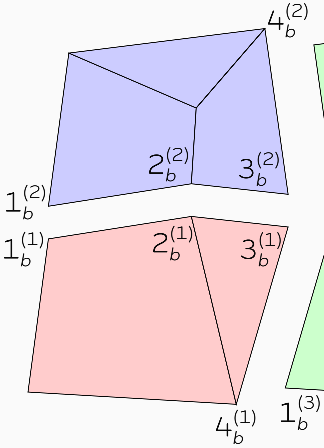


[Gosselet and Rey 2006]

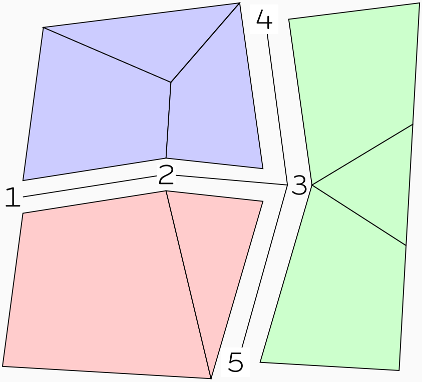


Elimination of interior d.o.f.

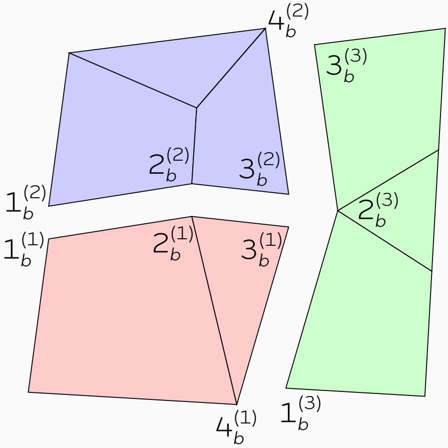
$$S^{(k)} = A_{bb} - A_{bi}A_{ii}^{-1}A_{ib}$$

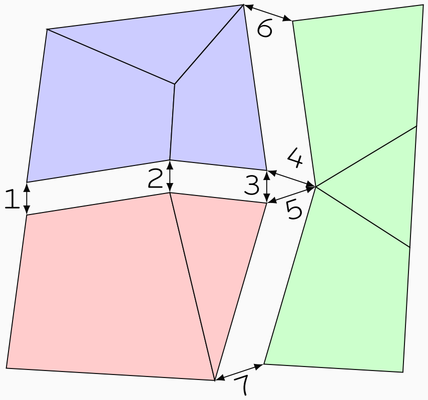


Jump operators: $\{B^{(i)}\}_{i=1}^{3}$



Primal constraints
[Mandel 1993]





Jump operators: $\{\underline{B}^{(i)}\}_{i=1}^{3}$

Dual constraints [Farhat and Roux 1991]

LIMITATIONS OF ONE-LEVEL METHODS

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

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

This hampers numerical scalability of such preconditioners:

$$\kappa(M^{-1}A) \leqslant C\frac{1}{H^2}\left(1+\frac{H}{\delta}\right)$$

- level of overlap δ,
- characteristic size of a subdomain H.

[Le Tallec 1994; Toselli and Widlund 2005]

TWO-LEVEL 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 $\mathcal{O}(N)$ columns, hence E is much smaller than A.

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

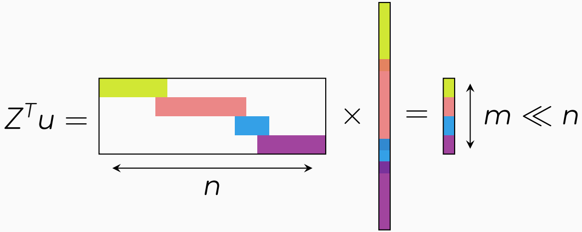
$$P^{-1} = M^{-1} + Z E^{-1} Z^T,$$

cf. [Tang et al. 2009].

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

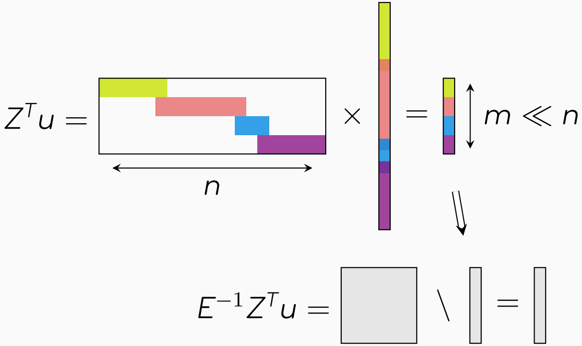
COMPUTING A COARSE OPERATOR CORRECTION

How to apply Z^T to $u \in \mathbb{K}^n$?



operations & `MPI_Gather`

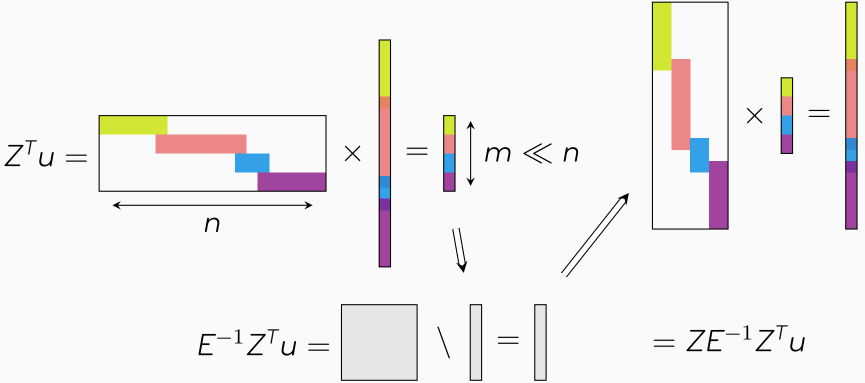
How to apply $E^{-1}Z^T$ to $u \in \mathbb{K}^n$?



operations & `MPI_Gather` + linear solve

COMPUTING A COARSE OPERATOR CORRECTION

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



operations & `MPI_Gather` + linear solve + `MPI_Scatter` & operations

NUMERICAL RESULTS

MACHINES USED FOR SCALING RUNS

Curie Thin Nodes

- 5 040 compute nodes (2 eight-core Intel Sandy Bridge).
- IB QDR full fat-tree.

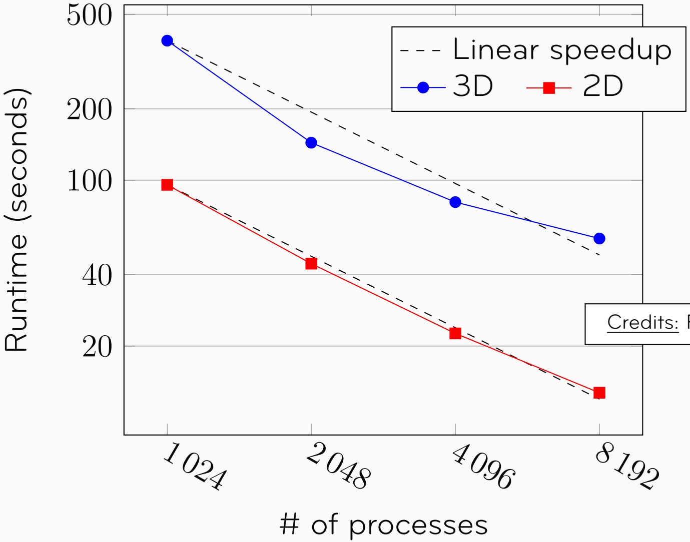
Turing

- 6 BlueGene/Q racks.

PRACE

GENCI
GRAND EQUIPEMENT NATIONAL DE CALCUL INTENSIF

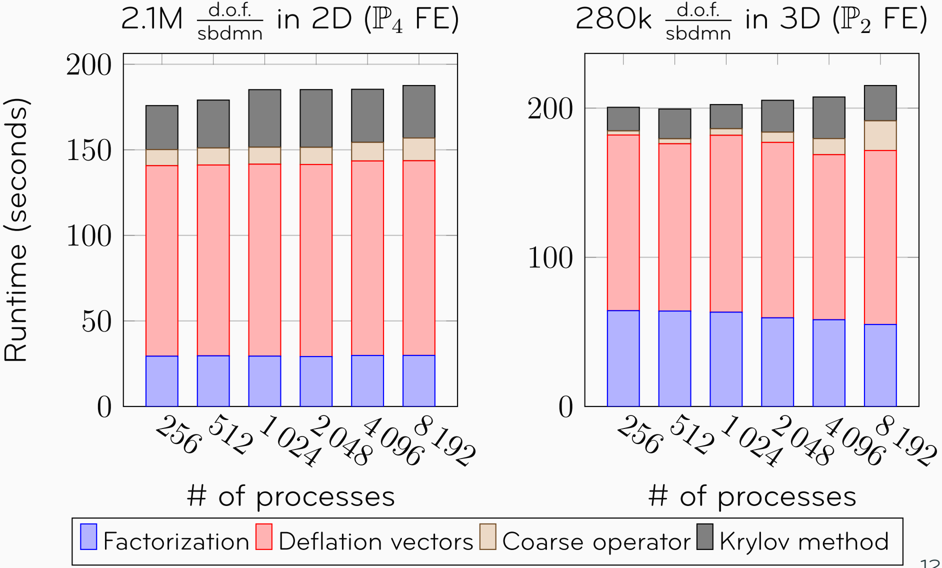
1 subdomain/MPI process, `--ranks-per-node` = 8.



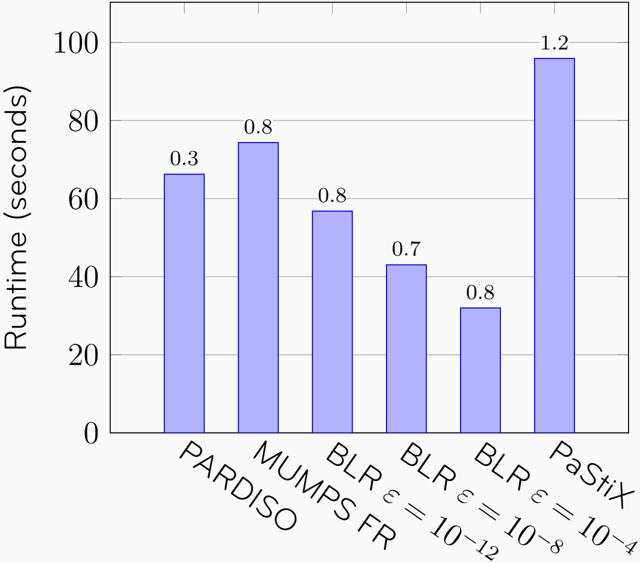
Credits: R. Haferssas

During setup, lot of time spent in the direct solver.

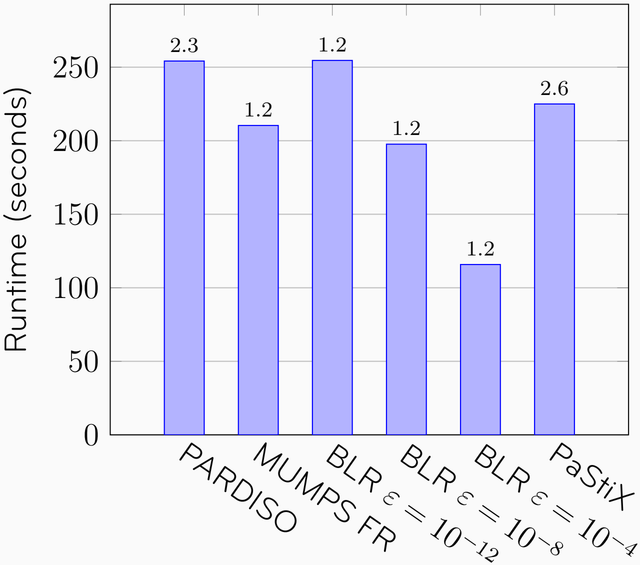
1 subdomain/MPI process, 2 OpenMP threads/MPI process.



LL^T factorization (430k d.o.f.)

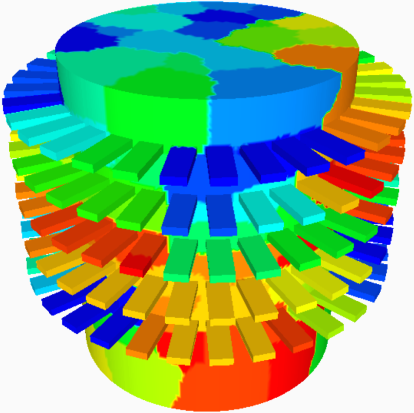


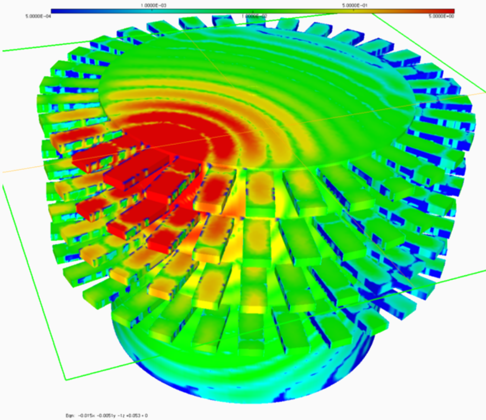
LDL^T factorization (503k d.o.f.)

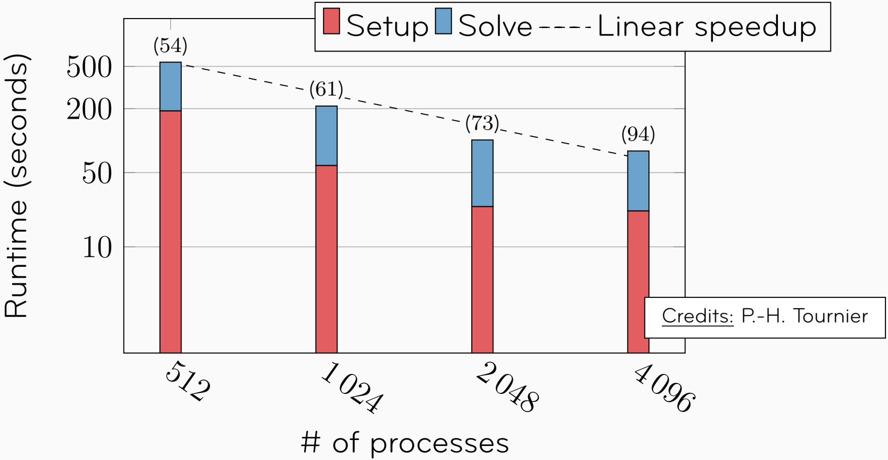


DD preconditioner for tomographic imaging.

$$\nabla \times (\nabla \times \mathbf{E}) - \mu_0 \Big(\omega^2 \varepsilon + \mathrm{i}\omega\sigma \Big) \mathbf{E} = 0$$

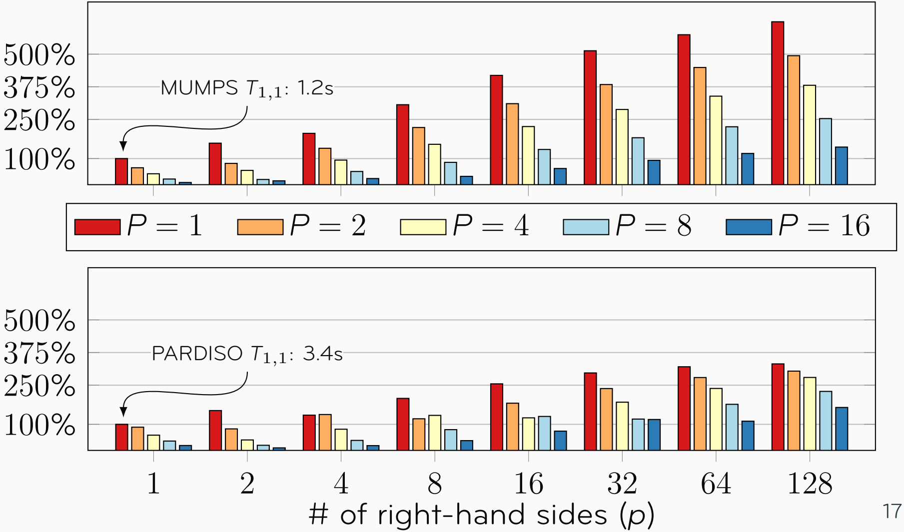






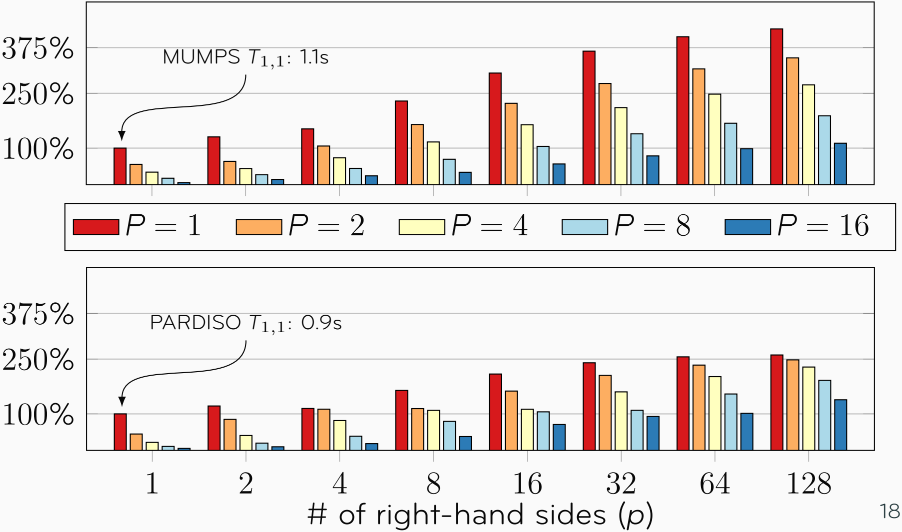
- 119 million double-precision complex unknowns
- degree 2 edge elements

$$E_{P,p} = \frac{p \cdot T_{1,1}}{P \cdot T_{P,p}}$$



MAXWELL'S EQUATION (LDL^T FACTORIZATION)

$$E_{P,p} = \frac{p \cdot T_{1,1}}{P \cdot T_{P,p}}$$



alternative	p	solve	# of it.	per RHS	eff.
GMRES	1				
GCRO-DR	1				

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2 048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3 078.4	20 068	627	–
GCRO-DR	1	1 836.9	10 701	334	1.7

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2 048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3 078.4	20 068	627	–
GCRO-DR	1	1 836.9	10 701	334	1.7
BGMRES	32	724.8	158	–	4.2

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2 048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3 078.4	20 068	627	–
GCRO-DR	1	1 836.9	10 701	334	1.7
BGMRES	32	724.8	158	–	4.2
BGCRO-DR	8	677.6	524	131	4.5

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2 048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3 078.4	20 068	627	–
GCRO-DR	1	1 836.9	10 701	334	1.7
BGMRES	32	724.8	158	–	4.2
BGCRO-DR	8	677.6	524	131	4.5
BGCRO-DR	32	992.3	127	–	3.1

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2 048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3 078.4	20 068	627	–
GCRO-DR	1	1 836.9	10 701	334	1.7
BGMRES	32	724.8	158	–	4.2
BGCRO-DR	8	677.6	524	131	4.5
BGCRO-DR	32	992.3	127	–	3.1

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2 048 subdomains and 2 threads per subdomain
- alternative #1 to #5 $\implies$ 158$\times$ fewer iterations

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3 078.4	20 068	627	–
GCRO-DR	1	1 836.9	10 701	334	1.7
BGMRES	32	724.8	158	–	4.2
BGCRO-DR	8	677.6	524	131	4.5
BGCRO-DR	32	992.3	127	–	3.1

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2 048 subdomains and 2 threads per subdomain
- alternative #1 to #5 $\implies$ 158$\times$ fewer iterations
- working on all 32 RHSs is costly (#4 vs. #5)

FEATURES AND WHISH LIST

SUMMARY OF USED FEATURES

- LU, LDL^T
- distributed or centralized solution
- BLR
- multiple RHS
- detection of pivots and size of nullspace
- computation of Schur complement
- working host/distributed input

WHISH LIST

- distributed RHS
- efficient handling of block matrices

- distributed RHS
- efficient handling of block matrices

Thank you for MUMPS
and for your attention!