Parallel algebraic domain decomposition linear solvers

HiePACS - Inria Bordeaux Sud-Ouest

First Brazil-France workshop
Sophia Antipolis, July 25-27 2012

Outline

1. Parallel Hybrid solver based on algebraic domain decomposition
   - HIPS solver
   - MaPHys solver
2. A geometric full mutigrid solver
3. Block Krylov linear solver

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- Robust/prescribed accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems
- Limited weak scalability

Iterative

- Problem dependent efficiency / monitored accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster
- Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- Robust/prescribed accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems
- Limited weak scalability

Iterative

- Problem dependent efficiency / monitored accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster
- Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- Robust/prescribed accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems
- Limited weak scalability

Iterative

- Problem dependent efficiency / monitored accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster
- Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- Robust/prescribed accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems
- Limited weak scalability

Iterative

- Problem dependent efficiency / monitored accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster
- Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- Robust/prescribed accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems
- Limited weak scalability

Iterative

- Problem dependent efficiency / monitored accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster
- Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- Robust/prescribed accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems
- Limited weak scalability

Iterative

- Problem dependent efficiency / monitored accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster
- Possible high weak scalability

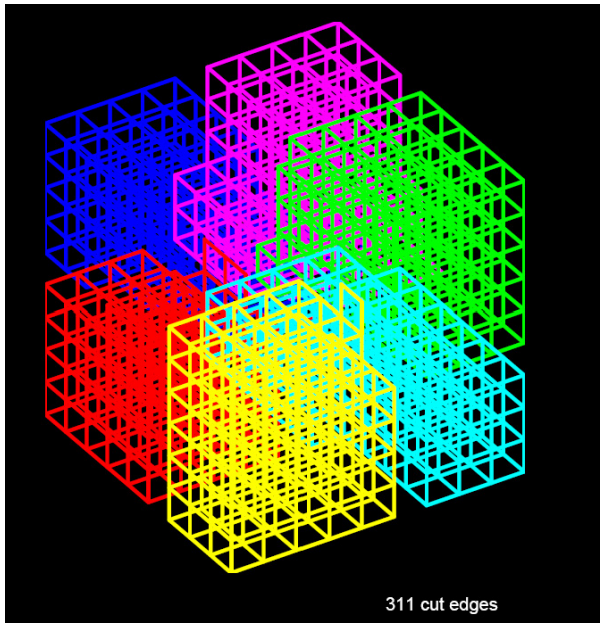
Governing Ideas in Hybrid Linear Solvers

Develop robust scalable parallel hybrid direct/iterative linear solvers

- Exploit the efficiency and robustness of the sparse direct solvers
- Develop robust parallel preconditioners for iterative solvers
- Take advantage of the natural scalable parallel implementation of iterative solvers

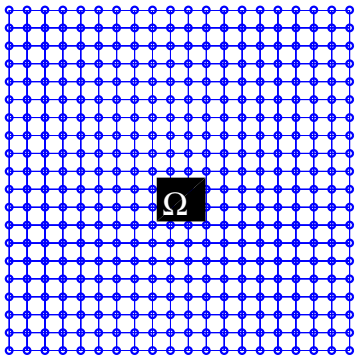
Domain Decomposition (DD)

- Natural approach for PDE's
- Extend to general sparse matrices
- Partition the problem into subdomains, subgraphs
- Use a direct solver on the subdomains
- Robust preconditioned iterative solver

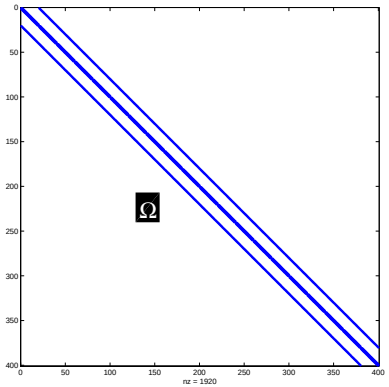


General partitioning of sparse matrix

Mesh view



Matrix view

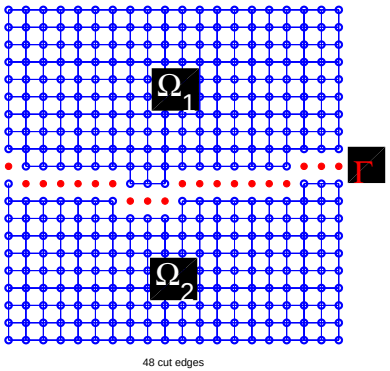


Tree view

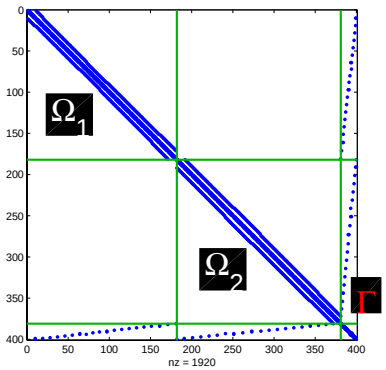
- Partitioning a matrix using algebraic algorithm based on the adjacency graph of $\mathcal{A}$
- 2 ways partitioning:
  - Computing an edge separator then finding the best vertex separator
  - Computing a vertex separator

General partitioning of sparse matrix

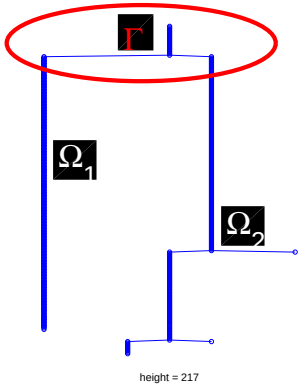
Mesh view



Matrix view



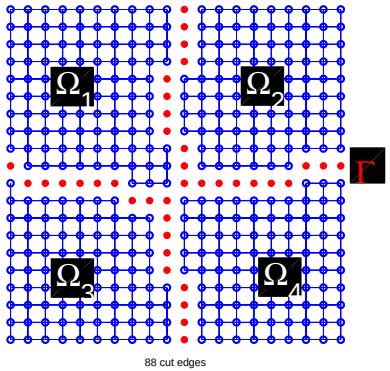
Tree view



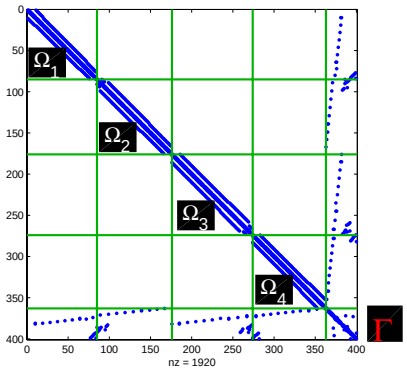
- Partitioning a matrix using algebraic algorithm based on the adjacency graph of $\mathcal{A}$
- 2 ways partitioning:
  - Computing an edge separator then finding the best vertex separator
  - Computing a vertex separator

General partitioning of sparse matrix

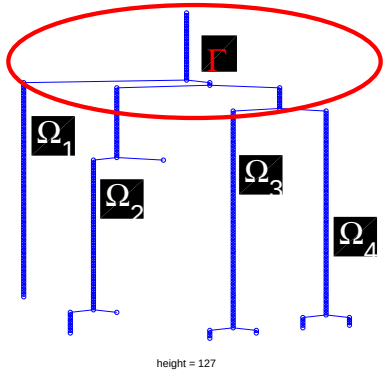
Mesh view



Matrix view



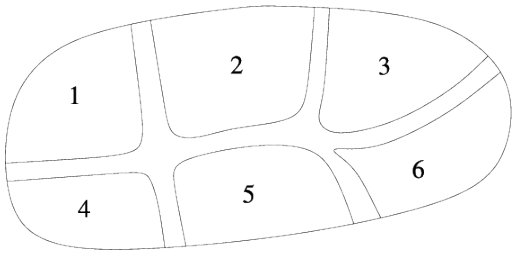
Tree view



HIPS : hybrid direct-iterative solver

Based on a domain decomposition : interface one node-wide (no overlap in DD lingo)

$$\begin{pmatrix} A_B & F \\ E & A_C \end{pmatrix}$$



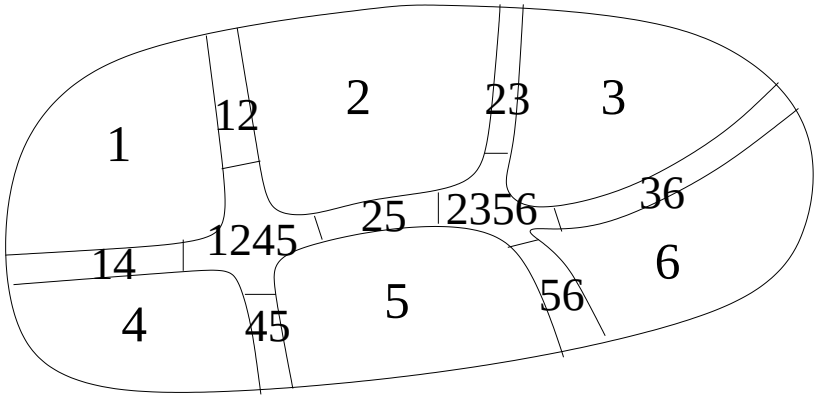
B : Interior nodes of subdomains (direct factorization).

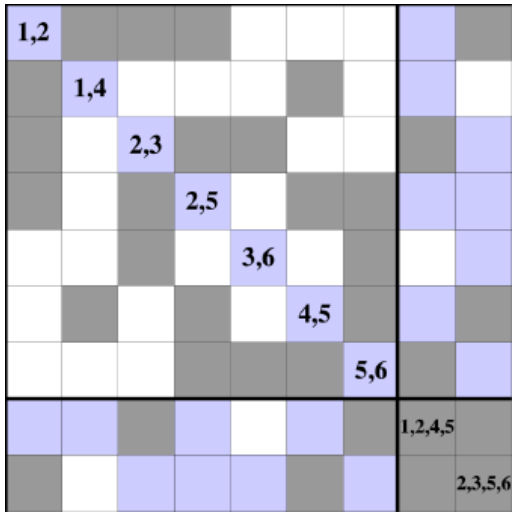
C : Interface nodes.

Special decomposition and ordering of the subset C :
Goal : Building a global Schur complement preconditioner (ILU) from the local domain matrices only.

HIPS: domain interface based fill-in policy

[P.Hénon, Y. Saad - SIAM SISC 06] [J.Gaidamour, P.Hénon -IEEE CSE 08]



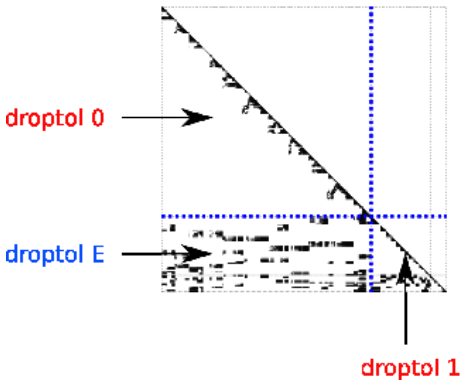


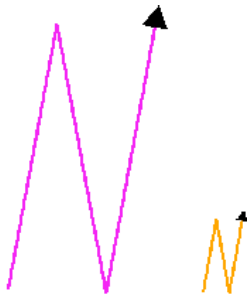
Robust block incomplete factorization of the Schur complement

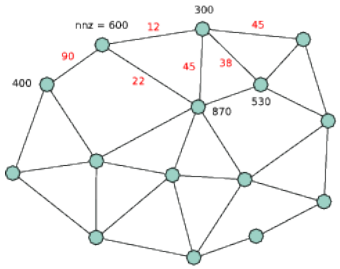
- Hierachy of separators (wirebasket like - faces , edges, vertices)
- Block incomplete factorization with “geometrical” fill-in policy to express parallelism
  (Global factorization using only local sub-domain matrices)
- MIS ordering to express parallelism within incomplete factorisation steps

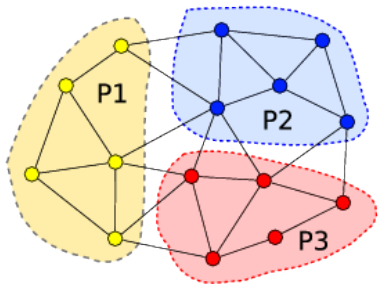
HIPS: preconditioners

[P.Hénon, Y. Saad - SIAM SISC 06] [J.Gaidamour, P.Hénon -CSE IEEE 08]









Main features

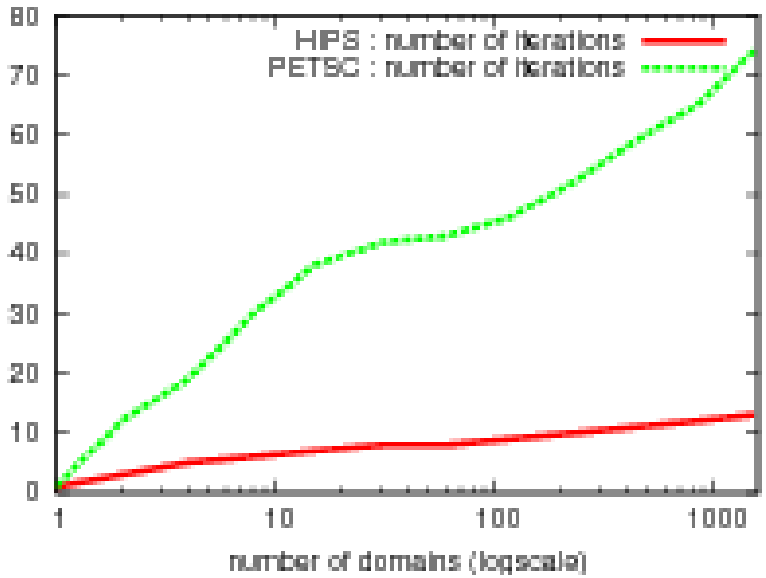
- Iterative or “hybrid” direct/iterative method are implemented.
- Mix direct supernodal (BLAS-3) and sparse ILUT factorization in a seamless manner.
- Memory/Load balancing : distribute the domains on the processors (domains $>$ processors).

HIPS vs Additive Schwarz (from PETSc)

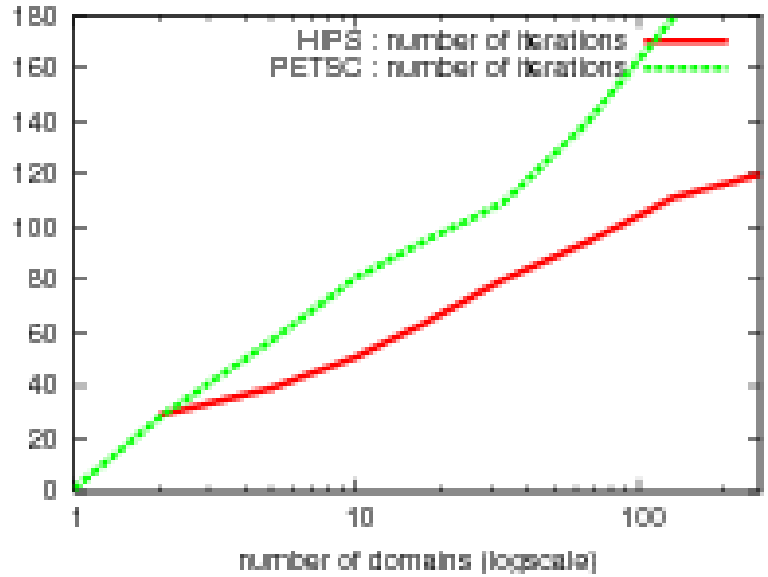
Experimental conditions

These curves compare HIPS (Hybrid) with Additive Schwarz from PETSc.
Parameters were tuned to compare the result with a very similar fill-in

Haltere



MHD



Hybrid solver : Amande up to 2048 cores on SGI Altix (Jade, CINES)

- Amandes : N=6, 994, 683 , NNZ=58, 477, 383
- Additive Schwarz, ILUT or ILUk failed
- 2053 domains of $\simeq$ 3770 nodes

Nb proc	Precond.	Solve	Total
1	803.12	104.87	907.99
2	384.12	58.84	442.95
4	205.96	46.87	252.83
8	129.35	21.00	150.35
16	65.50	18.81	84.31
32	35.15	9.42	44.57
64	18.51	4.79	23.31
128	9.84	2.41	12.25
256	5.84	1.41	7.26
512	3.80	0.69	4.49
1024	3.44	0.38	3.82
2048	4.76	0.34	5.10

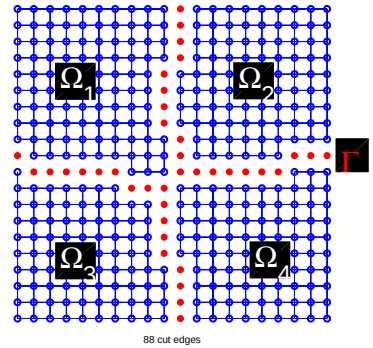
Global algebraic view

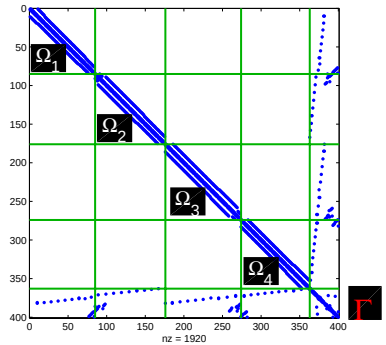
- Global hybrid decomposition:

$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}\mathcal{I}} & \mathcal{A}_{\mathcal{I}\Gamma} \\ \mathcal{A}_{\Gamma\mathcal{I}} & \mathcal{A}_{\Gamma\Gamma} \end{pmatrix}$$

- Global Schur complement:

$$\mathcal{S} = \mathcal{A}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma\mathcal{I}}\mathcal{A}_{\mathcal{I}\mathcal{I}}^{-1}\mathcal{A}_{\mathcal{I}\Gamma}$$





Local algebraic view

- Local hybrid decomposition:

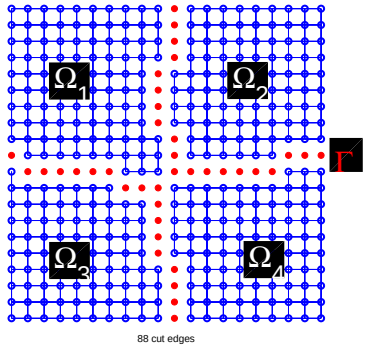
$$\mathcal{A}^{(i)} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}_i\mathcal{I}_i} & \mathcal{A}_{\mathcal{I}_i\Gamma_i} \\ \mathcal{A}_{\Gamma_i\mathcal{I}_i} & \mathcal{A}^{(i)}_{\Gamma\Gamma} \end{pmatrix}$$

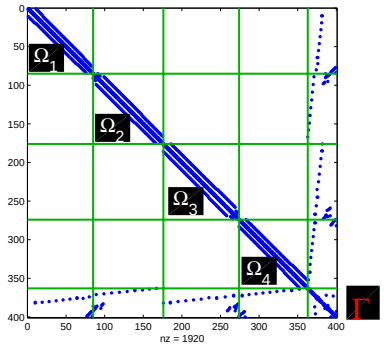
- Local Schur Complement:

$$\mathcal{S}^{(i)} = \mathcal{A}^{(i)}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma_i\mathcal{I}_i}\mathcal{A}^{-1}_{\mathcal{I}_i\mathcal{I}_i}\mathcal{A}_{\mathcal{I}_i\Gamma_i}$$

- Algebraic Additive Schwarz Preconditioner

$$\mathcal{M} = \sum_{i=1}^{N} \mathcal{R}^T_{\Gamma_i}(\bar{\mathcal{S}}^{(i)})^{-1}\mathcal{R}_{\Gamma_i}$$





Algebraic Additive Schwarz Preconditioner [L.Carvalho, L.Giraud, G.Meurant 01]

$$\mathcal{M} = \sum_{i=1}^{N} \mathcal{R}_{\Gamma_i}^T (\bar{\mathcal{S}}^{(i)})^{-1} \mathcal{R}_{\Gamma_i}$$

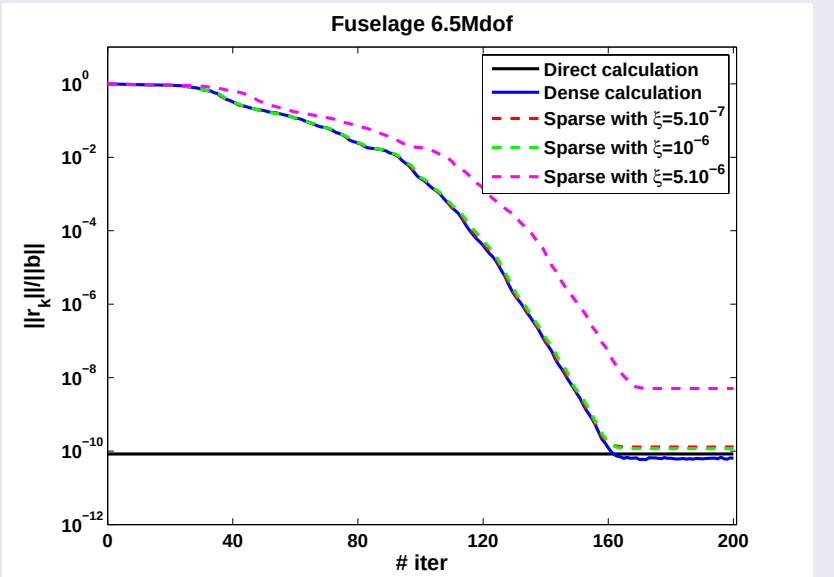
where $\bar{\mathcal{S}}^{(i)}$ is obtained from $\mathcal{S}^{(i)}$ via neighbor to neighbor comm

$$\underbrace{\mathcal{S}^{(i)} = \begin{pmatrix} \mathcal{S}_{kk}^{(\iota)} & \mathcal{S}_{k\ell} \\ \mathcal{S}_{\ell k} & \mathcal{S}_{\ell\ell}^{(\iota)} \end{pmatrix}}_{\text{local Schur}} \Longrightarrow \underbrace{\bar{\mathcal{S}}^{(i)} = \begin{pmatrix} \mathcal{S}_{kk} & \mathcal{S}_{k\ell} \\ \mathcal{S}_{\ell k} & \mathcal{S}_{\ell\ell} \end{pmatrix}}_{\text{local assembled Schur}}$$

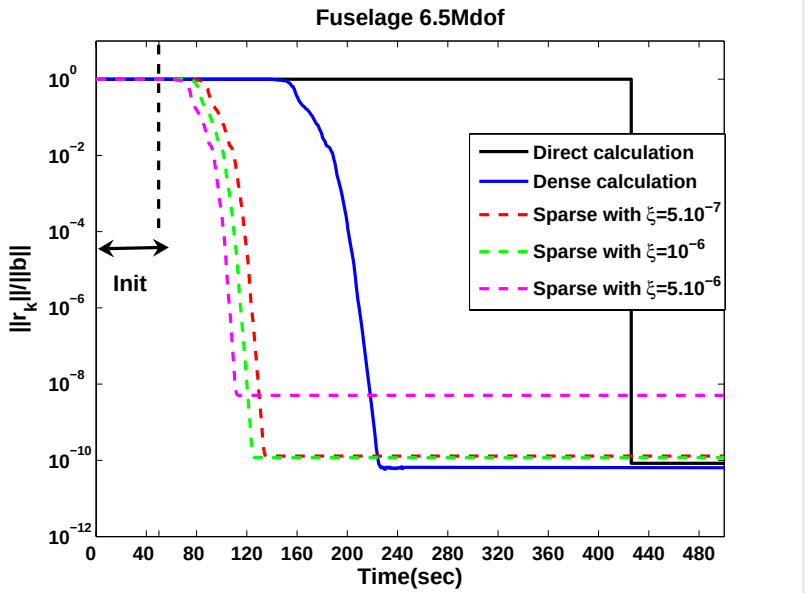
$$\sum_{\iota \in adj} \mathcal{S}_{\ell\ell}^{(\iota)}$$

Numerical behaviour of sparse preconditioners

Convergence history



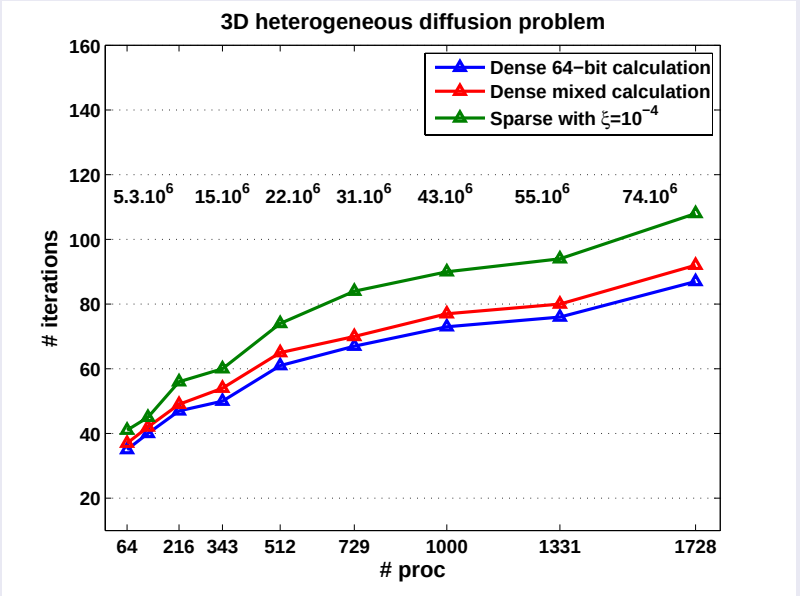
Time history



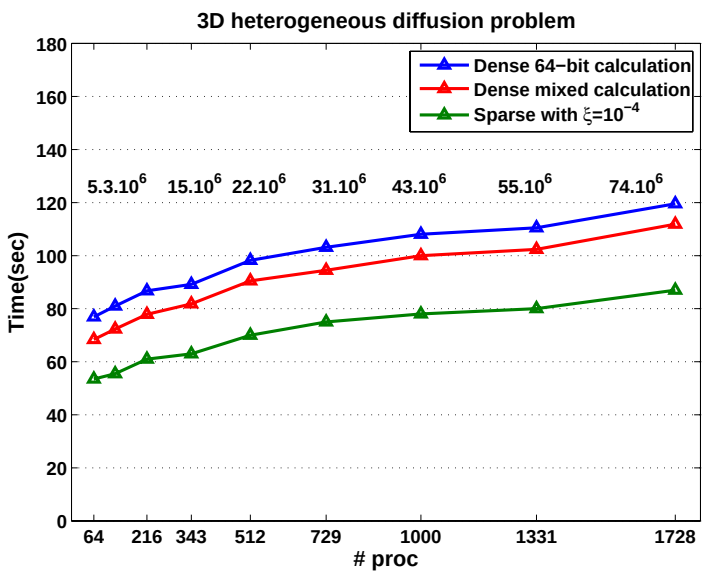
- Fuselage problem of 6.5 Mdof dof mapped on 16 processors
- The sparse preconditioner setup is 4 times faster than the dense one (19.5 v.s. 89 seconds)
- In term of global computing time, the sparse algorithm is about twice faster
- The attainable accuracy of the hybrid solver is comparable to the one computed with the direct solver

Scaled scalability on massively parallel platforms

Numerical scalability



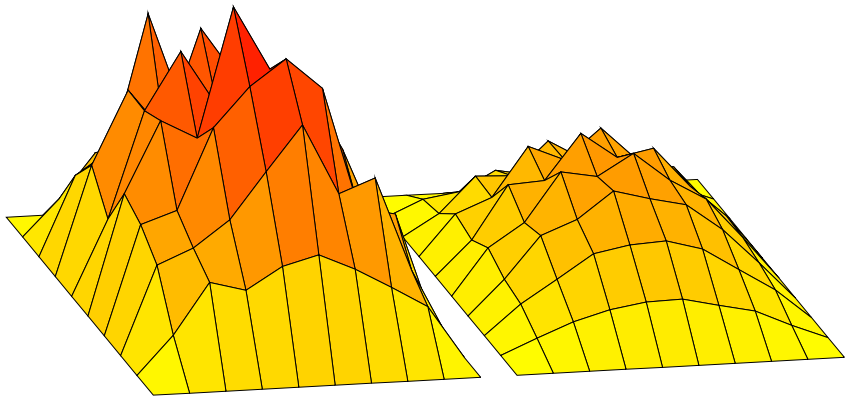
Parallel performance



- The solved problem size varies from 2.7 up to 74 Mdof
- Control the growth in the # of iterations by introducing a coarse space correction
- The computing time increases slightly when increasing # sub-domains
- Although the preconditioners do not scale perfectly, the parallel time scalability is acceptable
- The trend is similar for all variants of the preconditioners using CG Krylov solver

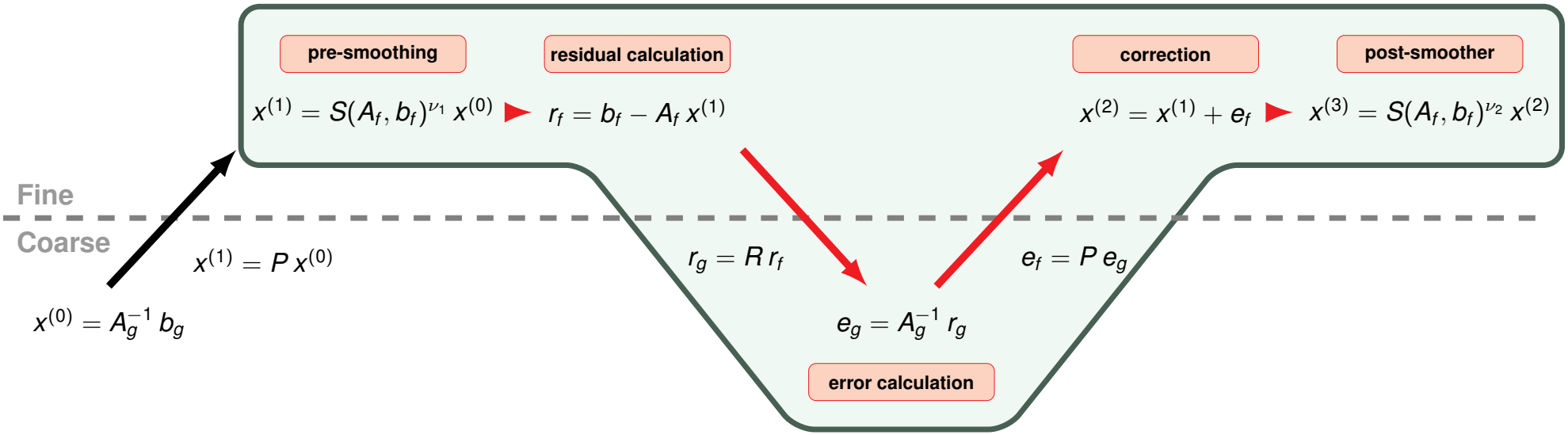
Multigrid Underlying ideas

- Stationary iteratve schemes versus the error modes
  - The high frequency modes are damped quickly
  - The low frequency modes are damped very slowly



- The low frequency modes might be viewed as high frequency on a coarser mesh

Full-Multigrid

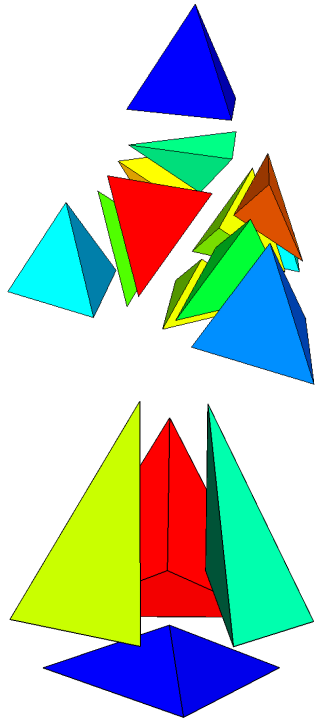


- Factorization of coarse problem performed only once
- Many forward/backward substitutions on coarsest grid
- Grid transfer operators to move within hierachical meshes
  - Fine $\rightarrow$ coarse : restriction
  - Coarse $\rightarrow$ fine : prolongation

Refinement/Coarsening strategies

Unstructured mesh refinement

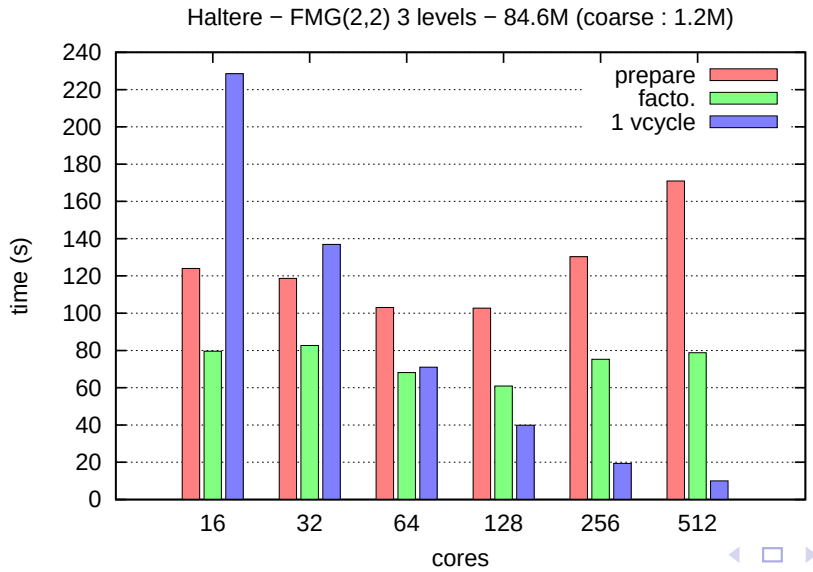
- Isotropic refinment
  - \+ Preserve aspect ration
  - – Large number of fine elements (x12)
  - – Coupling interface change
- Anisotropic refinement
  - \+ Coupling interface unchanged
  - \+ Slower increase of mesh size x4
  - – Bad aspect ratio



Strong scalability

Table: fine 84.6M – coarse 1.2 M – FMG(2,2) 3 levels

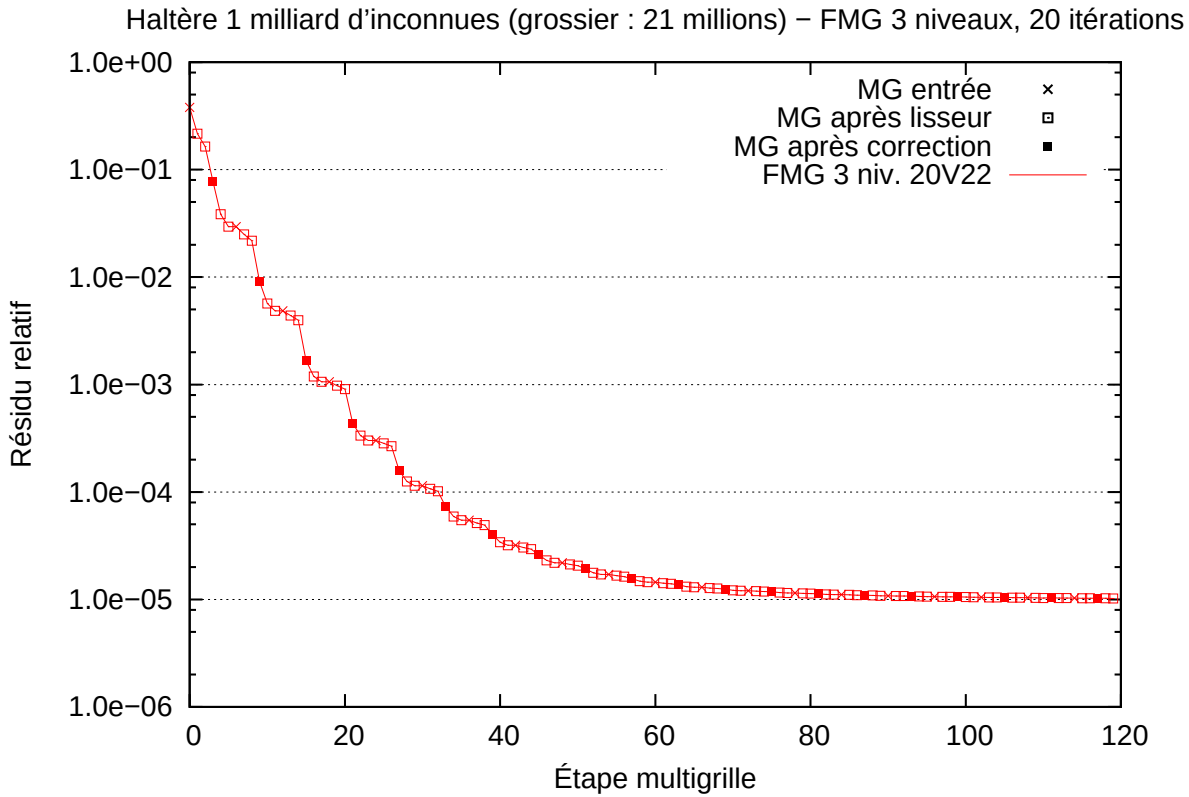
Cores	Init.	Assemb.	Facto.	Solve	10V
16	123.98	0.42	79.62	0.55	2285.58
32	118.67	0.23	82.72	0.30	1369.26
64	103.10	0.12	68.18	0.19	710.57
128	102.69	0.06	60.94	0.11	399.23
256	130.32	0.03	75.27	0.21	194.42
512	170.97	0.01	78.81	0.06	100.17



Validation – Strong scaling

Table: fine 1.3B – coarse 21.1 M – FMG(2,2) 3 levels

Cores	Init.	Assemb.	Facto.	Solve	10V
1024	1160.28	0.14	147.73	0.55	857.69



Block Krylov linear solver

We want to solve

$$AX = B,$$

where, $A \in \mathbb{C}^{N \times N}$, $B \in \mathbb{C}^{N \times p}$ full rank , and $X \in \mathbb{C}^{N \times p}$.

- For large sparse matrices, it might become prohibitive for sparse direct solvers because of high memory requirement and operation counts.

- Suited candidates are Block Krylov approaches:
  - The Krylov subspaces associated with each right-hand side are shared to enlarge the search subspace.
  - Matrix-vector products are simultaneously computed.

Background on Krylov subspace methods

$$Ax = b.$$

- The nested Krylov subspace of dimension m generated by A from the initial residual vector $r_0 = b - Ax_0$ is of the form

$$\mathscr{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0\}.$$

- Arnoldi's procedure is used to build an orthonormal basis of the Krylov subspace.
  - Arnoldi equality:

$$AV_m = V_m H_m + [0, \quad \omega_m]$$

  - Block Arnoldi recurrence formula:

$$A\mathscr{V}_m = \mathscr{V}_m \mathscr{H}_m + [0, \quad W_m]$$

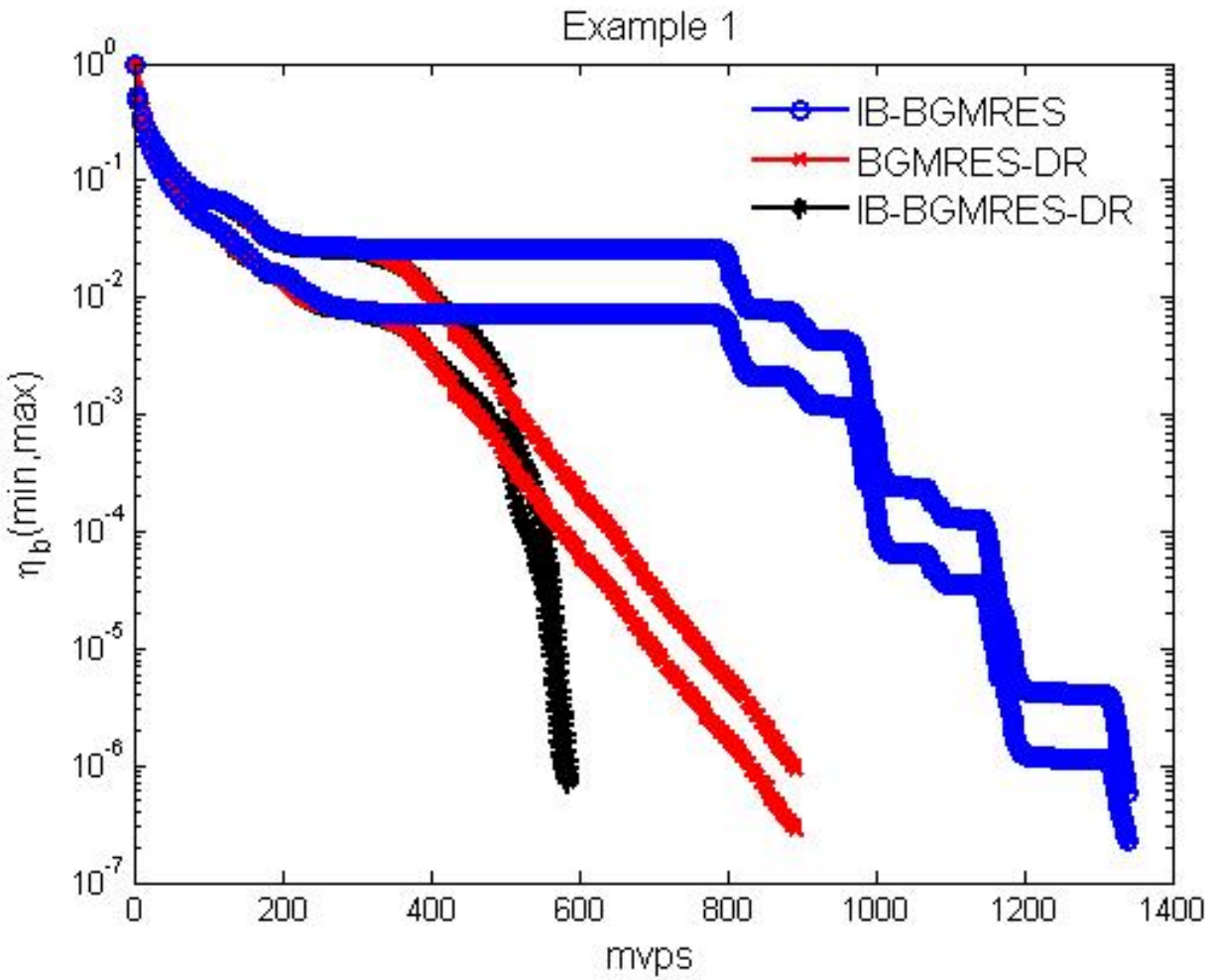
  - Block Arnoldi with inexact breakdown [Robbé, Sadkane - 06]

$$A\mathscr{V}_m = \mathscr{V}_m \mathscr{H}_m + [\mathfrak{L}_{m-1}, \quad W_m]$$

Motivation for this work

- GMRES with deflated restarting–GMRES-DR
  - Deflate targeted eigenvalues by combining their harmonic Ritz vectors at each restart to form the next seed Krylov subspace [R. B. Morgan. *GMRES with deflated restarting. SIAM J. Scientic Computing, 24(1):20-37, 2002*.
- Block variants
  - Block-GMRES-DR method by Morgan [R. B. Morgan, *Restarted block GMRES with deflation of eigenvalues*, Appl. Numer. Math., 54 (2005), pp. 222-236].
  - Block GMRES method by Robbé and Sadkane [M. Robbé and M. Sadkane, *Exact and inexact breakdowns in the block GMRES method*, Linear Algebra Appl., 419 (2006), pp. 265-285].

Preliminary numerical experiments



Block versus “regular” GMRES

Example	GMRES	GMRES-DR	IB-BGMRES	BGMRES-DR	IB-BGMRES-DR
1	2536	1077	1339	892	**587**
2	1069	856	787	667	**535**
3	378	378	372	341	**334**
4	**412**	**412**	444	447	439
5	845	694	617	474	**386**
6	464	464	357	294	**248**
7	3154	2003	3291	3090	**2104**
8	10643	3110	10000*	4426	**2197**

Table: Comparison results of regular GMRES, GMRES-DR, IB-BGMRES, BGMRES-DR and IB-BGMRES-DR in terms of matrix-vector products.

Ongoing work

- Hybrid solvers
  - `http://hips.gforge.inria.fr`
  - `https://wiki.bordeaux.inria.fr/maphys/doku.php`
- Design and implementation on heterogeneous manycore plateform
  PhD funded by Total/DIP, MORSE Associated team (ICL Tennessee, Kaust, UC Denver)
- Exploit H-matrix techniques to reduce computational resource comsumption
  FastLA Associated team (LBNL, Stanford)

Merci for your attention

Questions ?