Parallel solvers for linear equations

HiePACS project

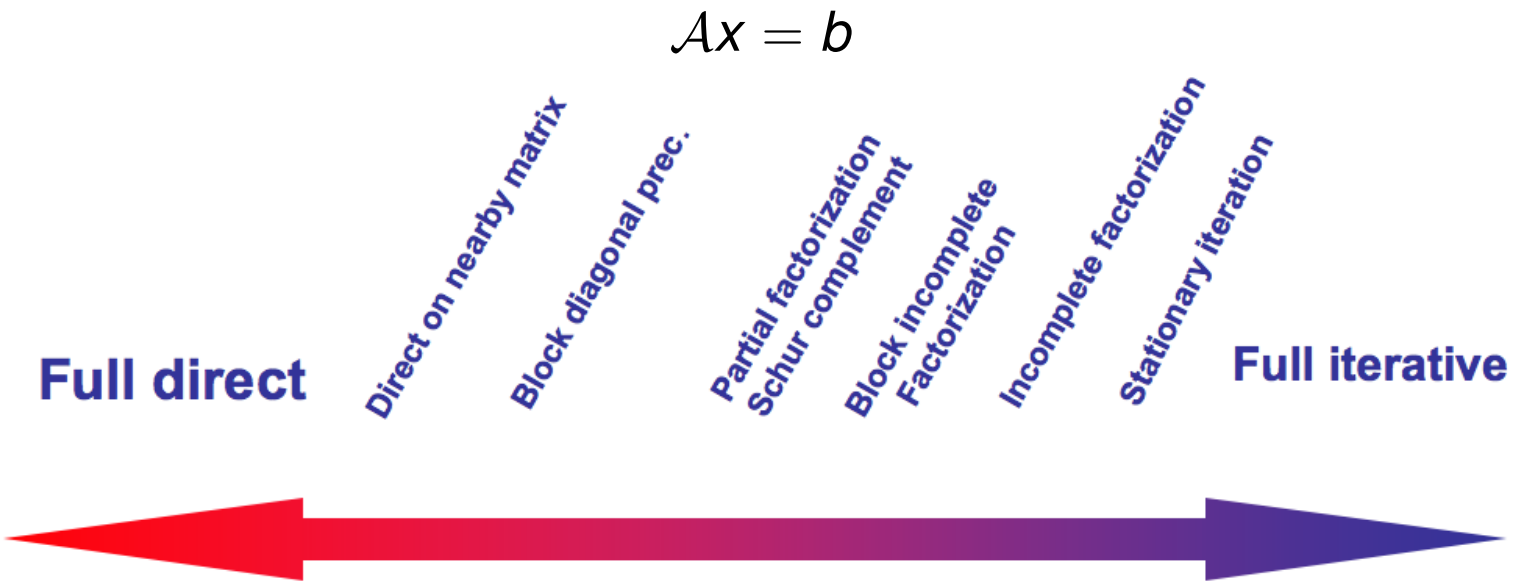
INRIA Bordeaux Sud-Ouest
joint INRIA-CERFACS lab. on High Performance Computing

Journe ANDRA C2S@Exa

Châtenay Malabry, April 2014

Outline

Motivations



The "spectrum" of linear algebra solvers

Direct

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

Iterative

- Problem dependent efficiency/controlled accuracy
- Only mat-vect required, fine grain computation
- Less memory computation, possible trade-off with CPU
- Attractive "build-in" parallel features

Sparse linear algebra @ Inria

Main Projects involved and related packages

- ALPINES
  CA-algorithms (LU, QR, ILU0), adaptive two level DDM for highly heterogeneous problems, parallel direction preserving preconditioners
- HiePACS
  dense linear algebra MORSE/MAGMA, sparse direct PaStiX, Hybrid (HIPS, MaPHyS)
- ROMA
  sparse direct MUMPS (in collaboration with INPT, CERFACS, Univ. de Bordeaux, CNRS, ENS Lyon)
- SAGE
  Krylov solvers (DGMRES, AGMRES in PETSC); Algebraic BNN (SIDNUR)

Outline

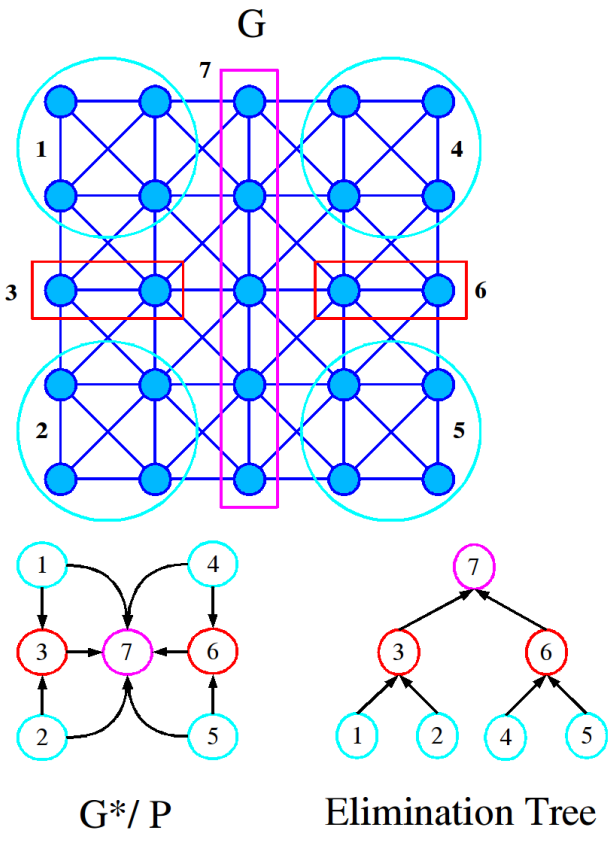
Parallel sparse direct solver - PaStiX Features

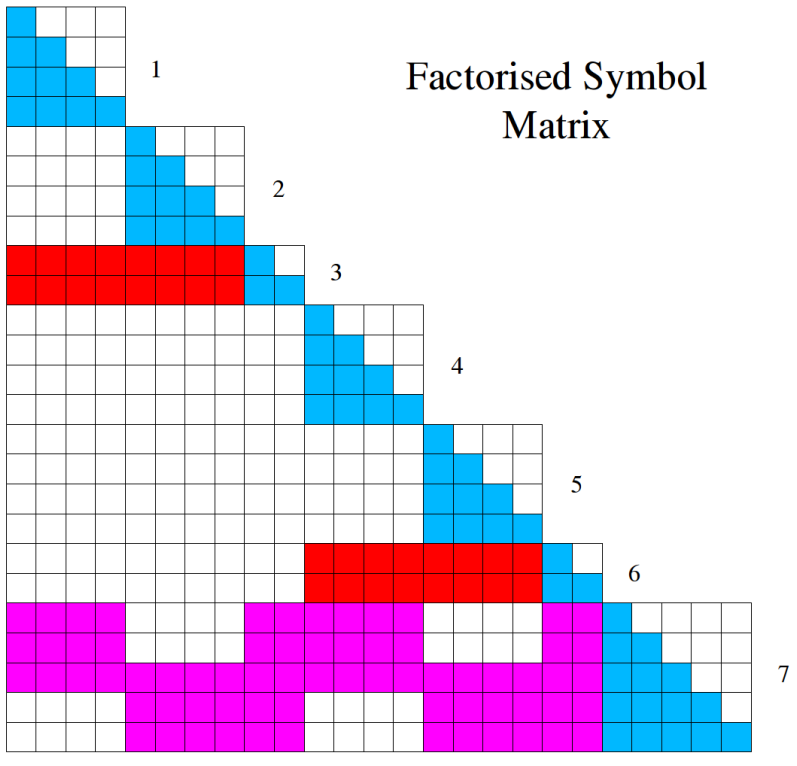
- LL^T, LDL^T, LU factorization with supernodal implementation
- Static pivoting + Refinement: CG/GMRES
- 1D/2D block distribution + Full BLAS3
- Simple/Double precision + Float/Complex operations

- MPI/Threads implementation (SMP/Cluster/Multicore/NUMA)
- **Dynamic scheduling inside SMP nodes (static mapping)**
- Support external ordering library (PT-Scotch/METIS)

- Multiple RHS (direct factorization)
- **Incomplete factorization with ILU(k) preconditionner**
- **Schur complement computation**
- Out-of Core implementation (in SMP mode only)

Direct Method





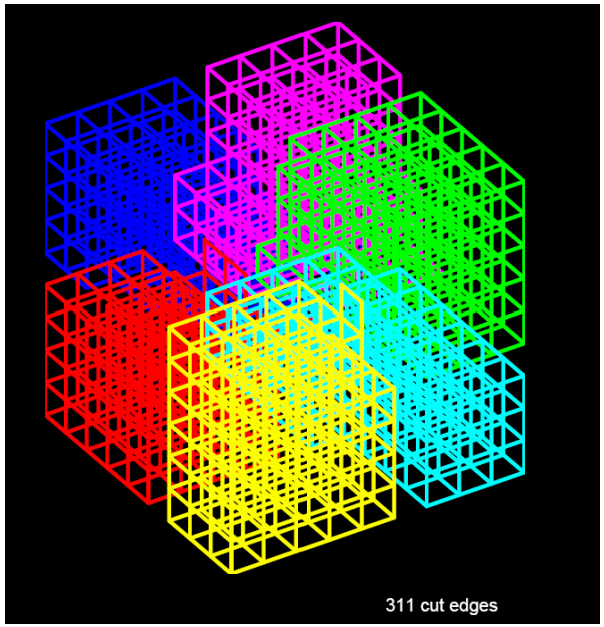
Hybrid Linear Solvers

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

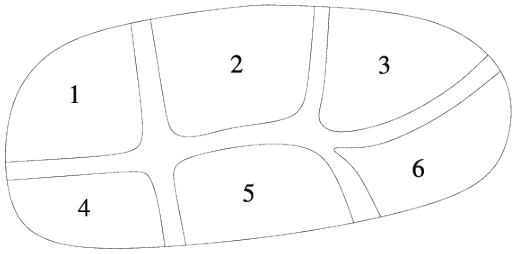


Outline

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.

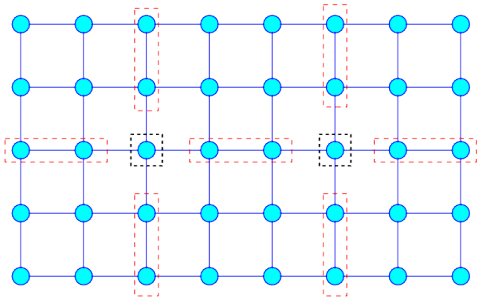
Special decomposition and ordering of the subset C :
Goal : Building a global Schur complement preconditioner (ILU) from the local domain matrices only.

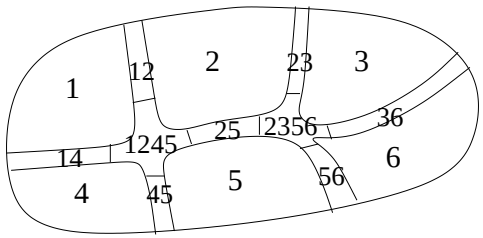
HIPS: domain interface based fill-in policy

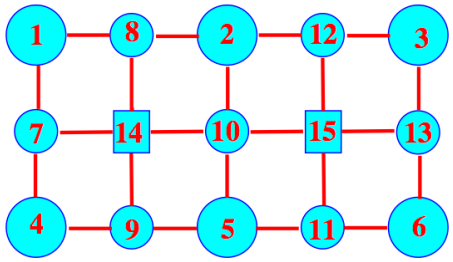
[P.Hénon, Y. Saad - SIAM SISC 06] [J.Gaidamour, P.Hénon -IEEE CSE 08]

Special decomposition and ordering of the subset C :

Hierachical interface decomposition into connectors :





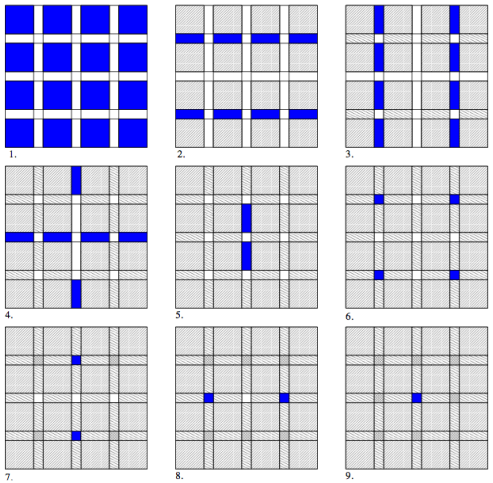


Rules :

- No creation of edge (fill-in) outside the local domain matrices.
- Allow edges between connectors adjacent to the same subdomain.

$\Rightarrow$ keep the parallelism (communication only between adjacent subdomains).

HID Elimination

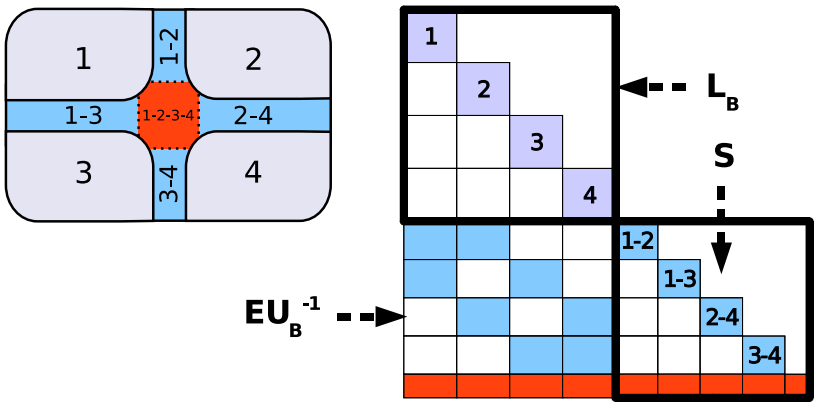


Robust block incomplete factorization of the Schur complement

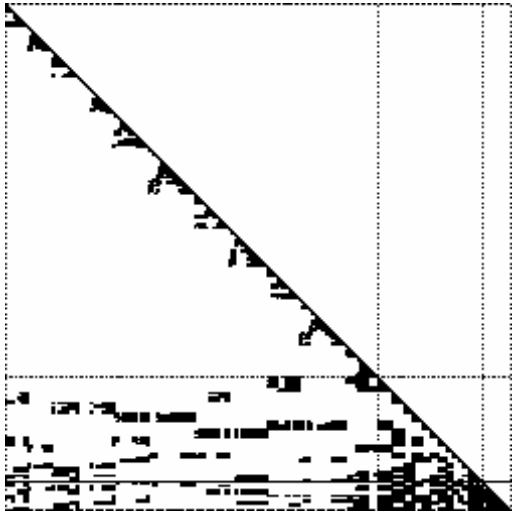
- Hierachy of separators (wirebasket like - faces , edges, vertices)
- Block incomplete factorization with “geometrical” fill-in policy to express parallelism
  (Global factorization using only local sub-domain matrices)
- MIS ordering to express parallelism within incomplete factorisation steps

Fill-in management policy

Matrix reordered according to the HID :



Symbolic factorization of the matrix BCSSTK14 :



$\Rightarrow$ the most part of the fill-in appear on EU_B^{-1}, $L_B^{-1}F$ and S (3D)

Reducing the memory footprint

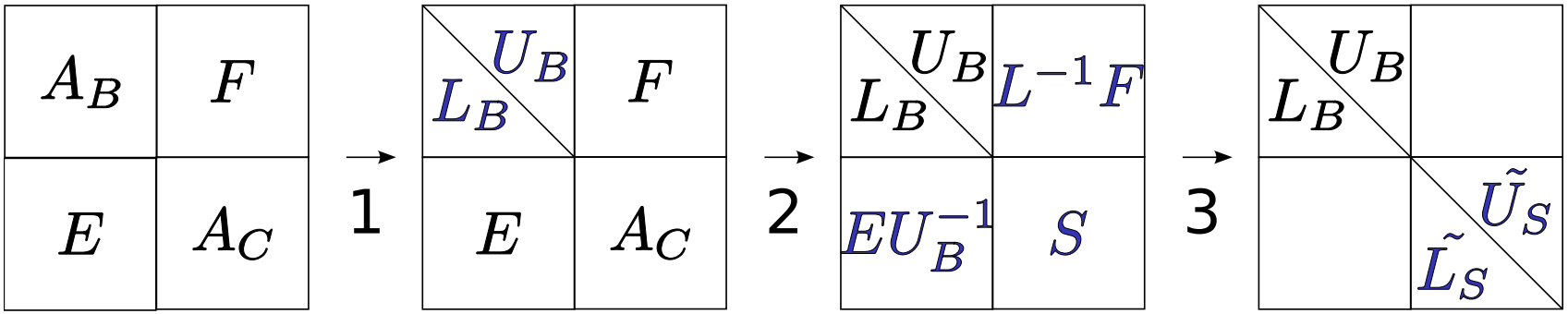
Objective : reduce the storage cost of EU_B^{-1}, $L_B^{-1}F$ and S ?

2 important remarks :

- The iterative resolution only needs the computation of $S.x$ (the Schur product). The iterative resolution only needs the computation of $S.x$ (the Schur product). It can be computed using $(A_C - EU_B^{-1}.L_B^{-1}F).x$.
- EU_B^{-1}, $L_B^{-1}F$ and S are only temporary matrices to compute $\tilde{L_S}.\tilde{U_S}$.

HIPS variant I

L_S, U_S factorization is based on an exact computation of S :



Main steps :

1. Exact factorization of $A_B = L_B.U_B$ (supernodal algorithm).
2. Computation of $W = EU_B^{-1}$, $G = L_B^{-1}F$ and the exact Schur S (supernodal right-looking algorithm).
3. ILU(τ_S) of S (scalar algorithm, τ_S is a numerical threshold).

How to avoid simultaneous storage of (W, G) and S ?

HIPS variant II

L_S, U_S factorization is based on an approximate computation of S :

$$S \simeq \tilde{S} \simeq \tilde{L_S}.\tilde{U_S}.$$

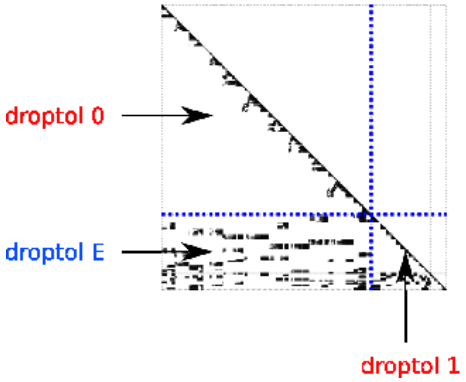
$\Rightarrow$ We accept to reduce the quality of the preconditioner to consume less memory.

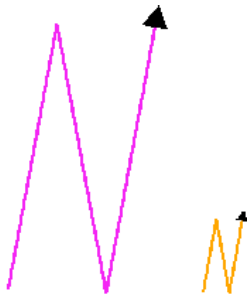
Main steps :

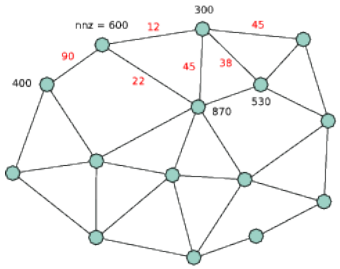
1. Exact factorization of $A_B = L_B.U_B$ (supernodal algorithm).
2. Approximate computation of $W = EU_B^{-1}$, $G = L_B^{-1}F$.
3. Left-Looking incomplete ILU(τ_S) factorization of $\tilde{S}$.

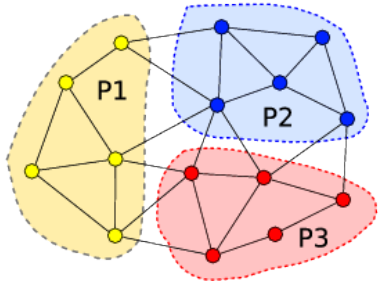
HIPS: preconditioners

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







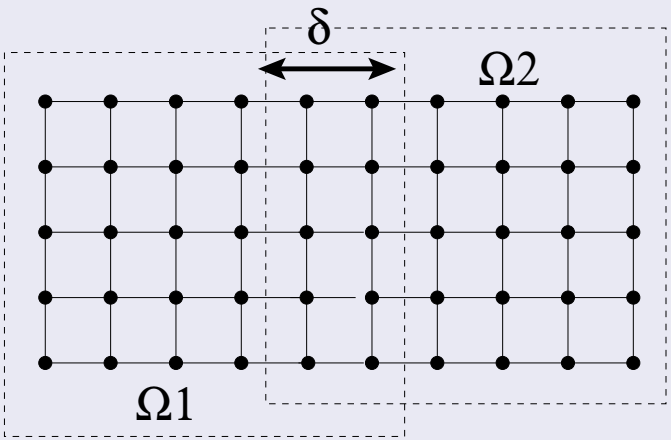


Main features

- 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).

Overlapping Domain Decomposition

Classical Additive Schwarz preconditioners



- *Goal:* solve linear system $\mathcal{A}x = b$
- Use iterative method
- Apply the preconditioner at each step
- The convergence rate deteriorates as the number of subdomains increases

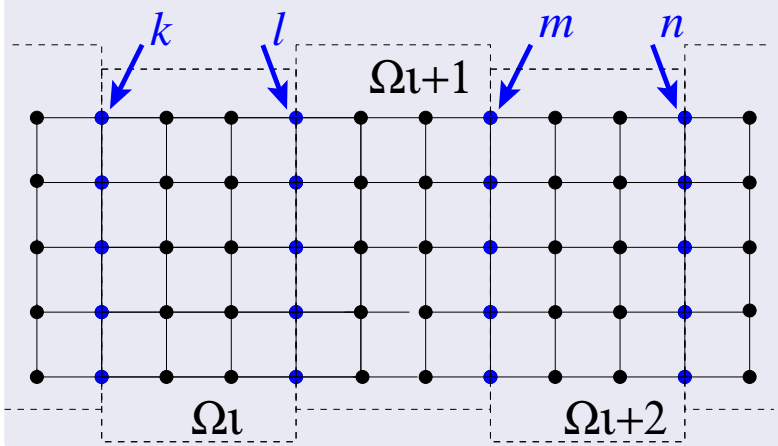
$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{1,1} & \mathcal{A}_{1,\delta} & \\ \mathcal{A}_{\delta,1} & \mathcal{A}_{\delta,\delta} & \mathcal{A}_{\delta,2} \\ & \mathcal{A}_{\delta,2} & \mathcal{A}_{2,2} \end{pmatrix} \Longrightarrow \mathcal{M}^{\delta}_{\mathcal{AS}} = \begin{pmatrix} \boxed{\begin{matrix} \mathcal{A}_{1,1} & \mathcal{A}_{1,\delta} \\ \mathcal{A}_{\delta,1} & \end{matrix}}^{-1} & & \\ & \boxed{\begin{matrix} \mathcal{A}_{\delta,\delta} & \mathcal{A}_{\delta,2} \\ \mathcal{A}_{\delta,2} & \mathcal{A}_{2,2} \end{matrix}}^{-1} & \end{pmatrix}$$

Classical Additive Schwarz preconditioners N subdomains case

$$\mathcal{M}^{\delta}_{AS} = \sum_{i=1}^{N} \left(\mathcal{R}^{\delta}_{i}\right)^{T} \left(\mathcal{A}^{\delta}_{i}\right)^{-1} \mathcal{R}^{\delta}_{i}$$

Nonoverlapping Domain Decomposition

Schur complement reduced system



$$\Gamma = k \cup \ell \cup m \cup n$$

Distributed Schur complement

$$\overbrace{\begin{pmatrix} \mathcal{S}_{kk}^{(\iota)} & \mathcal{S}_{k\ell} \\ \mathcal{S}_{\ell k} & \mathcal{S}_{\ell\ell}^{(\iota)} \end{pmatrix}}^{\Omega_\iota} \quad \overbrace{\begin{pmatrix} \mathcal{S}_{\ell\ell}^{(\iota+1)} & \mathcal{S}_{\ell m} \\ \mathcal{S}_{m\ell} & \mathcal{S}_{mm}^{(\iota+1)} \end{pmatrix}}^{\Omega_{\iota+1}} \quad \overbrace{\begin{pmatrix} \mathcal{S}_{mm}^{(\iota+2)} & \mathcal{S}_{mn} \\ \mathcal{S}_{nm} & \mathcal{S}_{nn}^{(\iota+2)} \end{pmatrix}}^{\Omega_{\iota+2}}$$

In an assembled form: $\mathcal{S}_{\ell\ell} = \mathcal{S}_{\ell\ell}^{(\iota)} + \mathcal{S}_{\ell\ell}^{(\iota+1)} \Longrightarrow \mathcal{S}_{\ell\ell} = \sum_{\iota \in adj} \mathcal{S}_{\ell\ell}^{(\iota)}$

Non-overlapping Domain Decomposition

Algebraic Additive Schwarz preconditioner [L.Carvalho, L.G., G.Meurant - 01]

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

$$\mathcal{S} = \begin{pmatrix} \ddots & & & & \\ & \mathcal{S}_{kk} & \mathcal{S}_{k\ell} & & \\ & \mathcal{S}_{\ell k} & \mathcal{S}_{\ell\ell} & \mathcal{S}_{\ell m} & \\ & & \mathcal{S}_{m\ell} & \mathcal{S}_{mm} & \mathcal{S}_{mn} \\ & & & \mathcal{S}_{nm} & \mathcal{S}_{nn} \end{pmatrix} \Longrightarrow \mathcal{M} = \begin{pmatrix} \ddots & & & & \\ & \mathcal{S}_{kk} & \mathcal{S}_{k\ell} & & \\ & \mathcal{S}_{\ell k} & \mathcal{S}_{\ell\ell} & \mathcal{S}_{\ell m} & \\ & & \mathcal{S}_{m\ell} & \mathcal{S}_{mm} & \mathcal{S}_{mn} \\ & & & \mathcal{S}_{nm} & \mathcal{S}_{nn} \end{pmatrix}^{-1}$$

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

where $\bar{\mathcal{S}}^{(i)}$ is obtained from $\mathcal{S}^{(i)}$

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

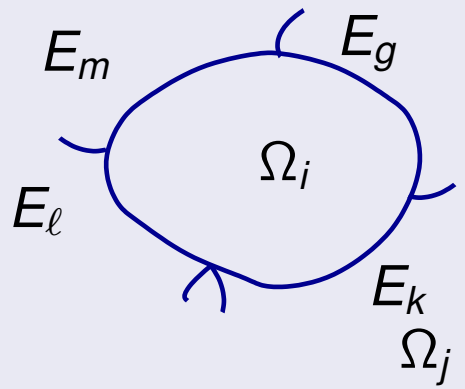
$$\sum_{\iota \in adj} \mathcal{S}_{\ell\ell}^{(\iota)}$$

Similarity with Neumann-Neumann preconditioner [J.F Bourgat, R. Glowinski, P. Le Tallec and M. Vidrascu - 89] [Y.H. de Roek, P. Le Tallec and M. Vidrascu - 91]

Parallel preconditioning features

$$S^{(i)} = A^{(i)}_{\Gamma_i\Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

$$M_{AS} = \sum_{i=1}^{\#domains} R_i^T (\bar{S}^{(i)})^{-1} R_i$$



$$\bar{S}^{(i)} = \begin{pmatrix} S_{mm} & S_{mg} & S_{mk} & S_{m\ell} \\ S_{gm} & S_{gg} & S_{gk} & S_{g\ell} \\ S_{km} & S_{kg} & S_{kk} & S_{k\ell} \\ S_{\ell m} & S_{\ell g} & S_{\ell k} & S_{\ell\ell} \end{pmatrix}$$

Assembled local Schur complement

$$S^{(i)} = \begin{pmatrix} S^{(i)}_{mm} & S_{mg} & S_{mk} & S_{m\ell} \\ S_{gm} & S^{(i)}_{gg} & S_{gk} & S_{g\ell} \\ S_{km} & S_{kg} & S^{(i)}_{kk} & S_{k\ell} \\ S_{\ell m} & S_{\ell g} & S_{\ell k} & S^{(i)}_{\ell\ell} \end{pmatrix}$$

local Schur complement

$$S_{mm} = \sum_{j \in adj(m)} S^{(j)}_{mm}$$

Parallel implementation

- Each *subdomain* $\mathcal{A}^{(i)}$ is handled by one *processor*

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

- Concurrent partial factorizations are performed on each processor to form the so called "local Schur complement"

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

- The reduced system $\mathcal{S}x_\Gamma = f$ is solved using a distributed Krylov solver
  - One matrix vector product per iteration each processor computes $\mathcal{S}^{(i)}(x^{(i)}_\Gamma)^k = (y^{(i)})^k$
  - One local preconditioner apply $(\mathcal{M}^{(i)})(z^{(i)})^k = (r^{(i)})^k$
  - Local neighbor-neighbor communication per iteration
  - Global reduction (dot products)
- Compute simultaneously the solution for the interior unknowns

$$\mathcal{A}_{\mathcal{I}_i\mathcal{I}_i}x_{\mathcal{I}_i} = b_{\mathcal{I}_i} - \mathcal{A}_{\mathcal{I}_i\Gamma_i}x_{\Gamma_i}$$

What tricks exist to construct cheaper preconditioners

Sparsification strategy through dropping

$$\widehat{s}_{k\ell} = \begin{cases} \bar{s}_{k\ell} & \text{if} \quad \bar{s}_{k\ell} \geq \xi(|\bar{s}_{kk}| + |\bar{s}_{\ell\ell}|) \\ 0 & \text{else} \end{cases}$$

Approximation through ILU - [INRIA PhyLeas - A. Haidar, L.G., Y.Saad - 10]

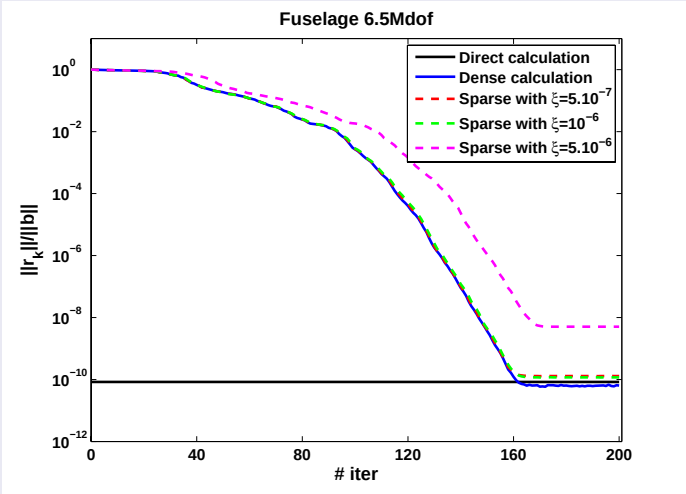
$$pILU\left(A^{(i)}\right) \equiv pILU \begin{pmatrix} A_{ii} & A_{i\Gamma_i} \\ A_{\Gamma_i i} & A^{(i)}_{\Gamma_i\Gamma_i} \end{pmatrix} \equiv \begin{pmatrix} \tilde{L}_i & 0 \\ A_{\Gamma i}\tilde{U}_i^{-1} & I \end{pmatrix} \begin{pmatrix} \tilde{U}_i & \tilde{L}_i^{-1}A_{i\Gamma} \\ 0 & \tilde{S}^{(i)} \end{pmatrix}$$

Mixed arithmetic strategy

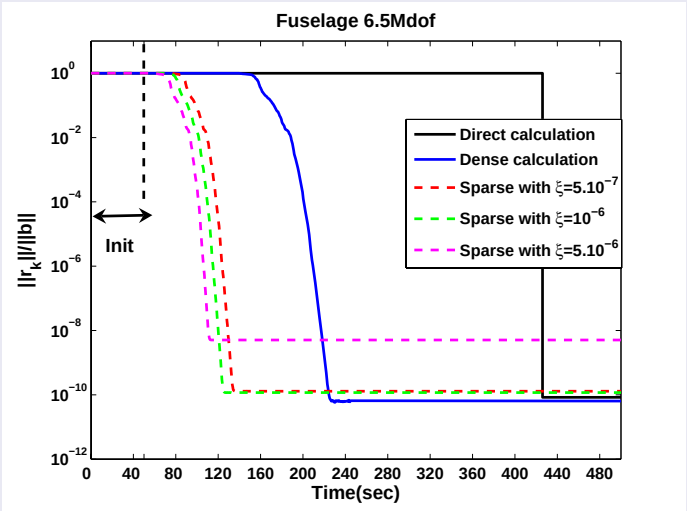
- Compute and store the preconditioner in 32-bit precision arithmetic Is accurate enough?
- Limitation when the conditioning exceeds the accuracy of the 32-bit computations Fix it!
- Idea: Exploit 32-bit operation whenever possible and ressort to 64-bit at critical stages
- Remarks: the backward stability result of GMRES indicates that it is hopeless to expect convergence at a backward error level smaller than the 32-bit accuracy [C.Paige, M.Rozložník, Z.Strakoš - 06]
- Idea: To overcome this limitation we use FGMRES [Y.Saad - 93]

Numerical behaviour of sparse preconditioners

Convergence history



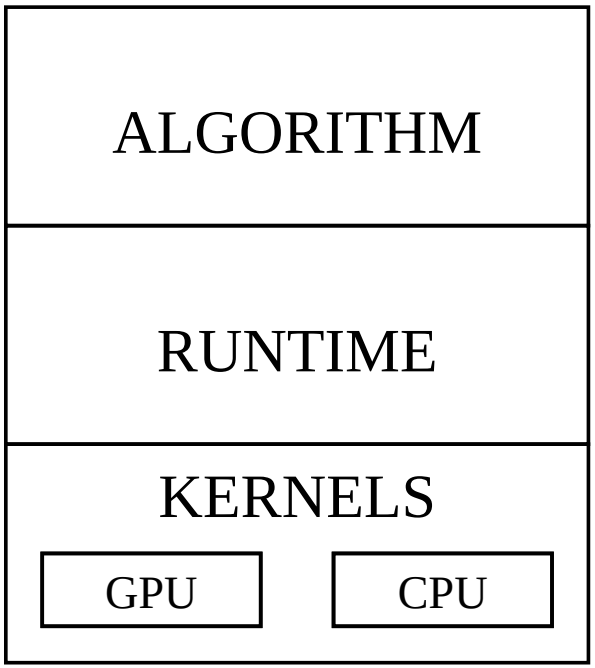
Time history



6.5 Mdof on 16 cores

Outline

Software approach : multiple layer approach

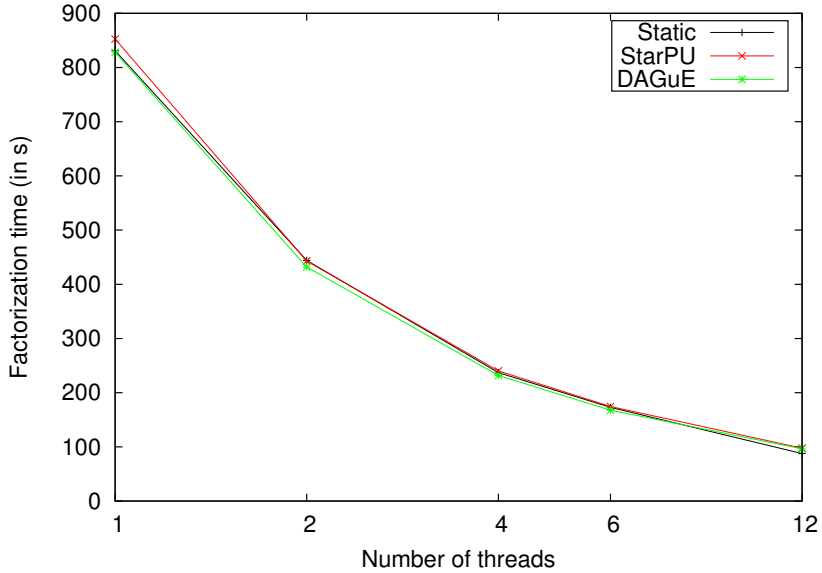


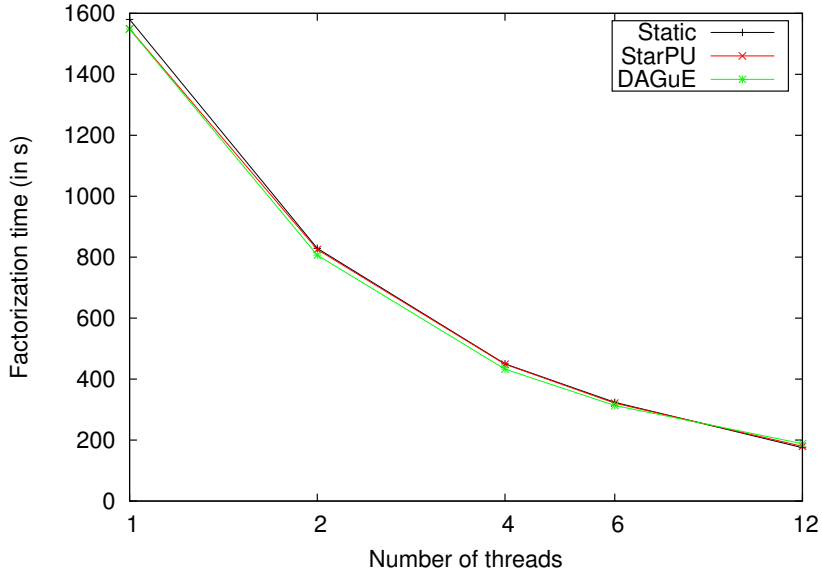
Governing ideas: Enable advanced numerical algorithms to be executed on a scalable unified runtime system for exploiting the full potential of future exascale machines.

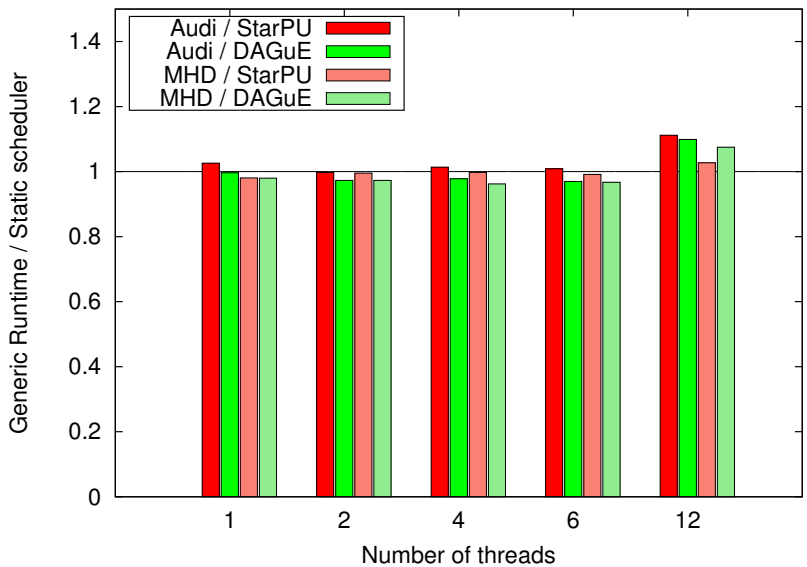
Basics:

- Graph of tasks
- Out-of-order scheduling
- Fine granularity

PaStiX : multicore results







PaStiX : results with GPUs over StarPU

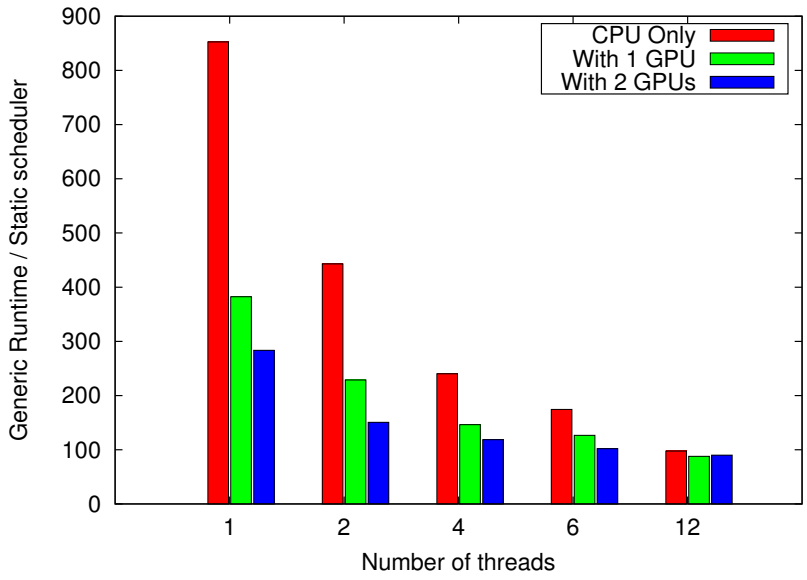


Figure: Audi

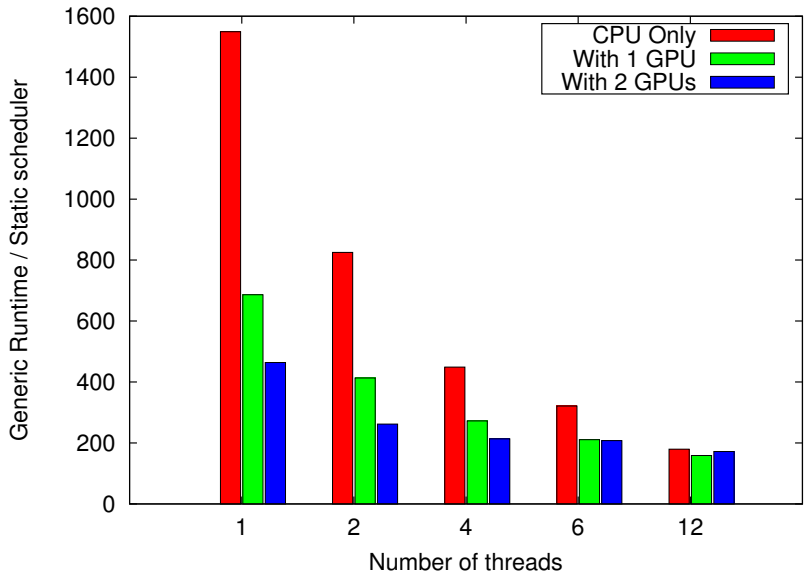


Figure: MHD