MaPHyS, a sparse hybrid linear solver and preliminary complexity analysis

Emmanuel AGULLO, Luc GIRAUD, Abdou GUERMOUCHE, Azzam HAIDAR, Jean ROMAN

INRIA Bordeaux Sud Ouest
CERFACS
Université de Bordeaux

3rd “Scheduling in Aussois” workshop, Aussois, France, June 2-4, 2010

Outline

Outline

Introduction: general High-Performance framework

Modern platforms

- ⋆ Massively multiprocessors and multicores
- ⋆ Hierarchical structure
- ⋆ Huge number of computational ressources
- ⋆ Heterogeneous ressources (a nodes may contain: multicores, GPUs,...)

Necessity to adapt/design (new) algorithms to efficiently exploit these platforms

New algorithmic constraints

- ⋆ How to achieve a high scalability with codes initially designed to run over “small” number of processors?
- ⋆ How can complex applications/algorithms handle the complex memory hierarchy and heterogeneity?

Introduction: Solving Large Linear Systems

Problem

- Given: very Large/Huge *ill*-conditioned sparse linear systems $\mathcal{A}x = b$
- Want : solve these linear systems efficiently

A few observations

- Industrial problems require thousands of CPU-hours and many Gigabytes of memory storage
- Industrial problems are also getting difficult for both direct and iterative methods

Motivations

$$Ax = b$$

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 Hybrid (direct/iterative) Linear Solvers

General Hybrid Linear Solvers

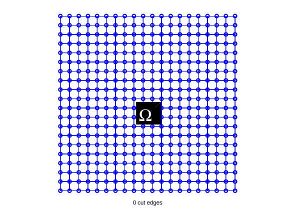
- ⋆ Given a matrix $\mathcal{A}$ or the adjacency graph of $\mathcal{A}$
- ⋆ Find independents sets of unknowns of $\mathcal{A}$ (partitioning or reordering), such that $\mathcal{A}$ can be written into that form:

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

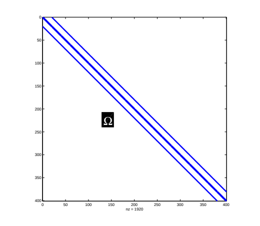
where $\mathcal{A}_{\mathcal{I}\mathcal{I}}$ is a block diagonal matrix forming an independent set of unknowns

General partitioning of sparse matrix

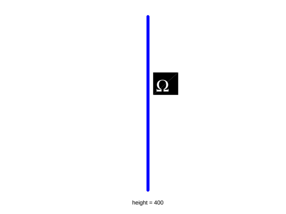
Mesh view



Matrix view



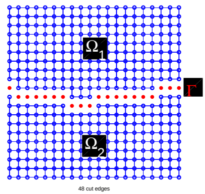
Tree view



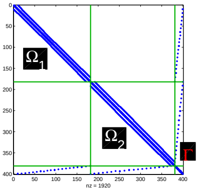
- ★ Partitioning a matrix using algebraic algorithm based on the adjacency graph of $\mathcal{A}$
- ★ No mesh will be used
- ★ 2 ways partitioning:
  - Computing an edge separator then finding the best vertex separator
  - Computing a vertex separator

General partitioning of sparse matrix

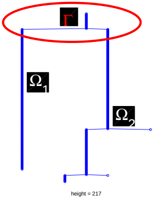
Mesh view



Matrix view



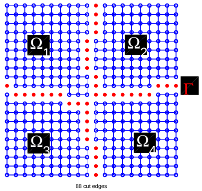
Tree view



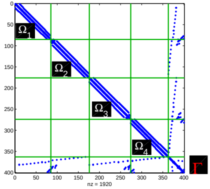
- ★ Partitioning a matrix using algebraic algorithm based on the adjacency graph of $\mathcal{A}$
- ★ No mesh will be used
- ★ 2 ways partitioning:
  - Computing an edge separator then finding the best vertex separator
  - Computing a vertex separator

General partitioning of sparse matrix

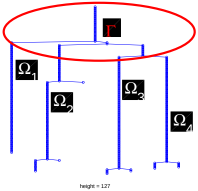
Mesh view



Matrix view



Tree view



- ⋆ Partitioners: METIS/PARMETIS, SCOTCH/PT-SCOTCH, ZOLTAN, PATOH ...
- ⋆ Recent trend: use of hypergraphs

Outline

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

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

Algebraic Additive Schwarz preconditioner

Brief description [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

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

local Schur ↘ $\sum_{\iota \in adj} \mathcal{S}_{\ell\ell}^{(\iota)}$ ↗ local assembled Schur

References

L. Giraud, A. Haidar, and L. T. Watson.
Parallel scalability study of hybrid preconditioners in three dimensions.
Parallel Computing, 34:363–379, 2008.

L. M. Carvalho, L. Giraud, and G. Meurant.
Local preconditioners for two-level non-overlapping domain decomposition methods.
Numerical Linear Algebra with Applications, 8(4):207–227, 2001.

What tricks exist to construct cheaper preconditioners

Sparsification strategy

- Sparsify the preconditioner by dropping the smallest entries

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

- Good in many PDE contexts
- Remarks: This sparse strategy was originally developed for SPD matrices

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]

Outline

Computational framework

Target computer

- ★ IBM SP4 @ CINES
- ★ Cray XD1 @ CERFACS
- ★ IBM JS21 @ CERFACS
- ★ Blue Gene/L @ CERFACS
- ★ IBM SP4 @ IDRIS
- ★ System X @ VIRGINIA TECH

System X @ VIRGINIA TECH

- ★ 2200 processors
- ★ Apple Xserve G5
- ★ 2-Way SMP
- ★ running at 2.3 GHz
- ★ 4 Gbytes/node
- ★ latency of 6.1 μs

Blue Gene/L @ CERFACS

- ★ 4096 processors
- ★ PowerPC 440s
- ★ 2-Way SMP
- ★ running at 700 MHz
- ★ 1 Gbytes/node
- ★ latency of 1.3 - 10 μs

IBM JS21 @ CERFACS

- ★ 216 processors
- ★ PowerPC 970MP
- ★ 4-Way SMP
- ★ running at 2.5 GHz
- ★ 8 Gbytes/node
- ★ latency of 3.2 μs

Software framework

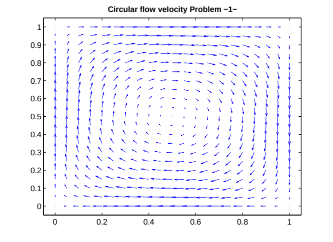
Software framework

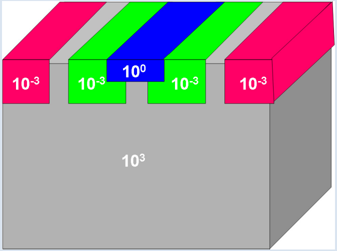
- ★ **METIS** G. Karypis and V. Kumar
  - Partitioning tool
  - Public domain:
    `http://glaros.dtc.umn.edu/gkhome/metis/metis/`
- ★ **MUMPS** P.Amestoy et al.
  - Local direct solver
  - Parallel distributed multifrontal solver
  - Public domain:
    `http://mumps.enseeiht.fr/`
- ★ **CG/GMRES/FGMRES** V.Frayssé, L.Giraud
  - Parallel distributed iterative solver
  - Public domain:
    `http://www.cerfacs.fr/algor/Softs/`

Outline

Academic model problems

Problem patterns





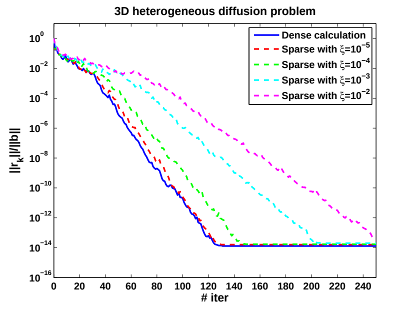
Diffusion equation $(\epsilon = 1$ and $v = 0)$ and convection-diffusion equation

$$\begin{cases} -\epsilon \text{div}(K.\nabla u) + v.\nabla u & = & f & \text{in} & \Omega, \\ u & = & 0 & \text{on} & \partial\Omega. \end{cases}$$

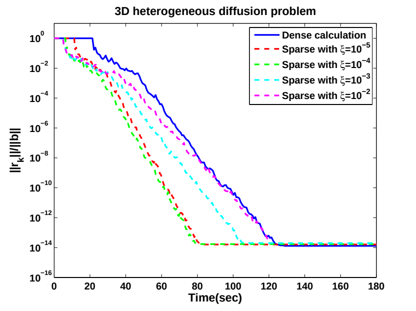
- ★ Heterogeneous problems
- ★ Anisotropic-heterogeneous problems
- ★ Convection dominated term

Numerical behaviour of sparse preconditioners

Convergence history of PCG



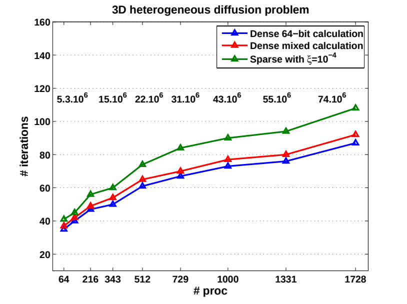
Time history of PCG



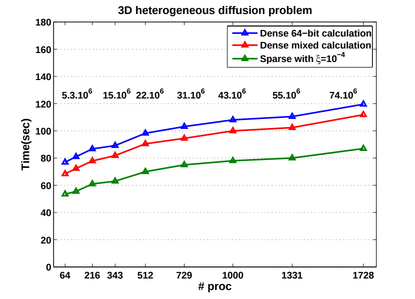
- ★ $3D$ heterogeneous diffusion problem with 43 Mdof mapped on 1000 processors
- ★ For ($\xi \lll$)the convergence is marginally affected while the memory saving is significant 15%
- ★ For ($\xi \ggg$) a lot of resources are saved but the convergence becomes very poor 1%
- ★ Even though they require more iterations, the sparsified variants converge faster as the time per iteration is smaller and the setup of the preconditioner is cheaper.

Weak scalability on massively parallel platforms

Numerical scalability



Parallel performance



- ★ The solved problem size varies from 2.7 up to 74 Mdof
- ★ Control the grow 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

Approximate Schur variant

Difficulties

- ★ The computation of the exact local Schur complement is expensive
- ★ Large amount of memory storage is required for very large applications

Approximate Schur variant of the preconditioner

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

where

$$\tilde{S}^{(i)} = A^{(i)}_{\Gamma_i\Gamma_i} - A_{\Gamma_i i}\tilde{U}_i^{-1}\tilde{L}_i^{-1}A_{i\Gamma_i}$$

Approximate Schur aproach: motivations joint work with Y. Saad

Exact vs. approximate Schur: memory saving (MB)

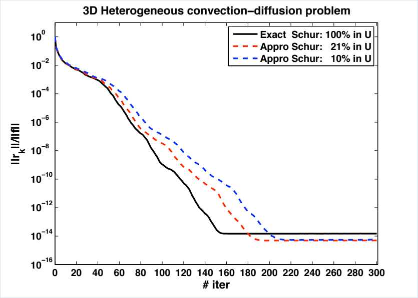
kept entries in factor	sub-domain mesh size						
	25^3 15 Kdof	30^3 27 Kdof	35^3 43 Kdof	40^3 64 Kdof	45^3 91 Kdof	50^3 125 Kdof	55^3 166 Kdof
Exact: 100% in U	254	551	1058	1861	3091	4760	7108
Appro: 21% in U	55	114	216	383	654	998	1506

Exact vs. approximate Schur: computing time (sec)

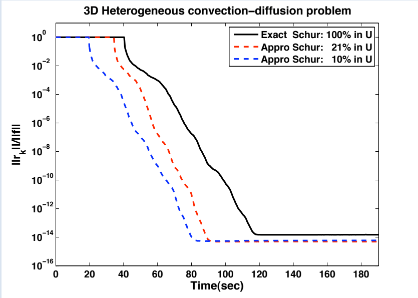
kept entries in factor	sub-domain grid size						
	25^3 15 Kdof	30^3 27 Kdof	35^3 43 Kdof	40^3 64 Kdof	45^3 91 Kdof	50^3 125 Kdof	55^3 166 Kdof
Exact: 100% in U	4.1	12.1	35.4	67.6	137	245	581
Appro: 21% in U	6.1	15.1	31.2	60.8	128	208	351
Appro: 10% in U	2.9	7.5	16.5	29.8	64	100	169

Numerical behaviour of approximate preconditioners

Convergence history of GMRES



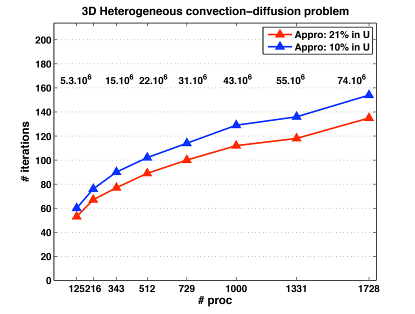
Time history of GMRES



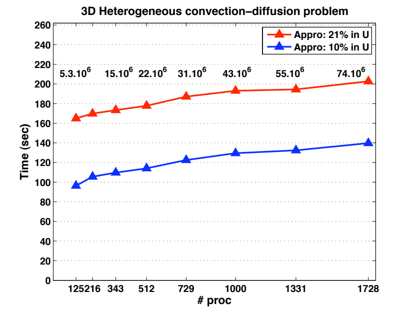
- ★ $3D$ heterogeneous convection-diffusion problem of 74 Mdof mapped on 1728 processors
- ★ the convergence is marginally affected while the memory saving is significant
- ★ Even though they require more iterations, the approximate variant converge faster as the time per iteration is smaller and the setup of the preconditioner is cheaper.

Weak scalability on massively parallel platforms

Numerical scalability



Parallel performance



- ★ The solved problem size varies from 2.7 up to 74 Mdof
- ★ The computing time increases slightly when increasing # sub-domains
- ★ Even if the number of iterations to converge increases as the number of subdomains increases, the parallel scalability of the preconditioners remains acceptable

Summary on the model problems

Ref: [L.Giraud, A.Haidar, L.T.Watson - 08] & [L.Giraud, A.Haidar, Y.Saad - 09]

Sparse preconditioner

- ★ For reasonable choice of the dropping parameter ξ the convergence is marginally affected
- ★ The sparse preconditioner outperforms the dense one in time and memory

Approximate preconditioner

- ★ The convergence is marginally affected while the memory saving is significant
- ★ The approximate variant converge faster as the time per iteration is smaller and the setup of the preconditioner is cheaper.
- ★ This preconditioner require some tuning for very hard problem (structural mechanics...)

On the weak scalability

- ★ Although these preconditioners are local, possibly not numerically scalable, they exhibit a fairly good parallel time scalability (possible fix for elliptic problems)
- ★ The trends that have been observed on this choice of model problem have been observed on many other problems

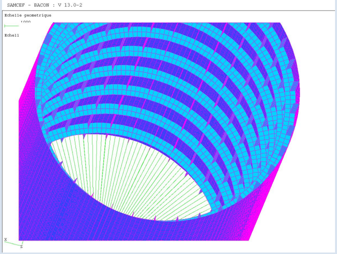
Outline

MaPHyS package

- ⋆ Study the performance of the hybrid solver in real life applications
- ⋆ Example: structural mechanics (indefinite system), electromagnetism, Helmholtz ...
- ⋆ Study the behavior of the preconditioner on more general problems
- ⋆ Black-box hybrid solver MaPHyS package

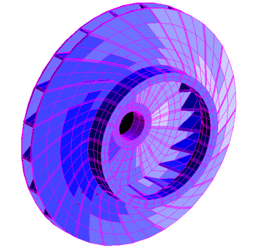
Indefinite systems in structural mechanics S. Pralet, SAMTECH

Fuselage of 6.5 Mdof



- ★ Composed of its skin, stringers and frames
- ★ Midlinn shell elements are used
- ★ Each node has 6 unknowns
- ★ A force perpendicular to the axis is applied

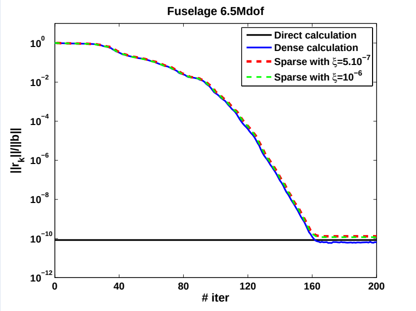
Rouet of 1.3 Mdof



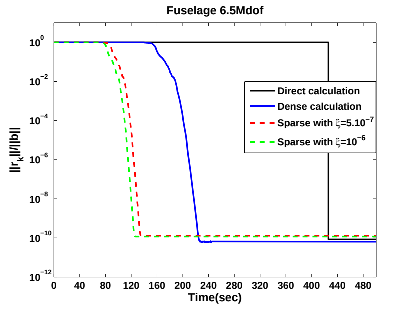
- ★ A 90 degrees sector of an impeller
- ★ It is composed of 3D volume elements
- ★ Cyclic conditions are added using elements with 3 Lagranges multipliers
- ★ Angular velocities are introduced

MaPHyS: numerical behaviour of the preconditioners

Convergence history



Time history

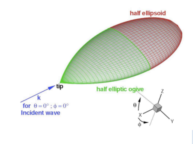


- ★ Fuselage problem of 6.5 Mdof dof mapped on 16 processors
- ★ The sparse preconditioner setup is 4 times faster than the dense one (19.5 v.s. 89 seconds)
- ★ In term of global computing time, the sparse algorithm is about twice faster
- ★ The attainable accuracy of the hybrid solver is comparable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

Black-box hybrid solver: problem characteristics

Amande (Almond) problem

- ★ Electromagnetism problem
- ★ 6,994,683 dof
- ★ 58,477,383 nnz



Haltere problem

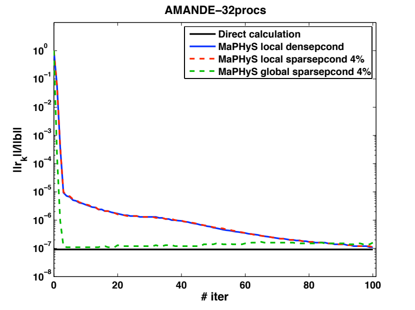
- ★ Electromagnetism problem
- ★ 1,288,825 dof
- ★ 10,476,775 nnz

Audi problem

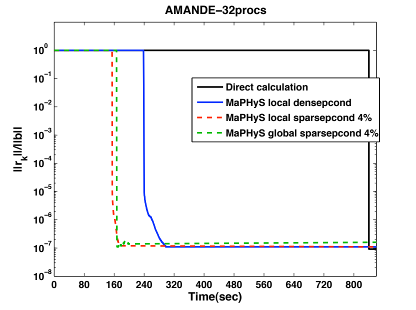
- ★ Structural mechanics problem
- ★ 943,695 dof
- ★ 39,297,771 nnz

MaPHyS: Amande problem

Convergence history



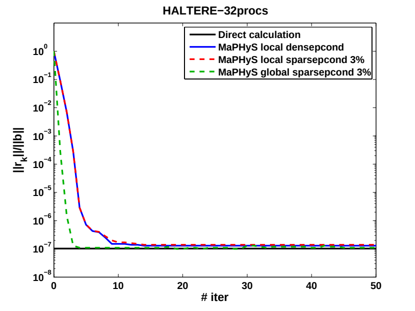
Time history



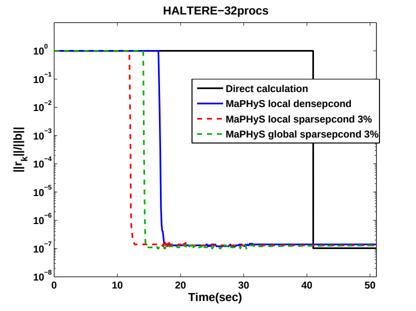
- ★ Amende problem of 6.99 Mdof mapped on 32 processors
- ★ In term of computing time, the sparse algorithm is about twice faster
- ★ The global sparse preconditioner perform very well on this number of processors
- ★ The attainable accuracy of the hybrid solver is comparable to the one computed with the direct solver

MaPHyS: Haltere problem

Convergence history



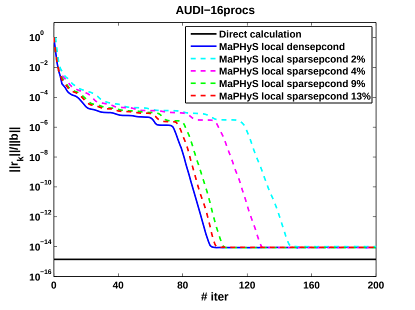
Time history



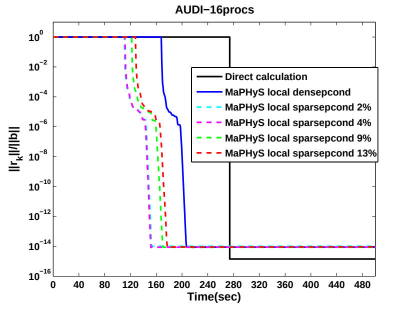
- ★ Haltere problem of 1.3 Mdof mapped on 32 processors
- ★ The local sparse algorithm perform as well as the dense
- ★ The global sparse preconditioner perform very well on this number of processors
- ★ The attainable accuracy of the hybrid solver is comparable to the one computed with the direct solver

MaPHyS: AUDI problem

Convergence history



Time history



- ★ Audi problem of 0.9 Mdof mapped on 16 processors
- ★ For ($\xi \lll$) the convergence is marginally affected while the memory saving is significant
- ★ For ($\xi \ggg$) a lot of resources are saved but the convergence becomes very poor
- ★ Even though they require more iterations, the sparsified variants performs faster
- ★ The attainable accuracy of the hybrid solver is comparable to the one computed with the direct solver

Outline

1. Introduction
2. Description of the hybrid approach
3. Experimental environment
4. Experiments on large $3D$ academic model problems
5. Experiments on large $3D$ real life applications
6. Preliminary complexity analysis
7. Perspectives

Framework

n^σ-separator theorem

A family of (structurally symmetric) matrices satisfies a n^σ-separator theorem when the graph associated to a matrix can be recursively partitionned as follows:

- ⋆ $G = (V, E)$, $|V| = n$;
- ⋆ $V = A \cup B \cup C$, C topological separator;
- ⋆ $|A|, |B| \leq \alpha n$, $C \leq \beta n^\sigma$;
- ⋆ $0 < \alpha < 1$, $\beta > 0$, $1/2 \leq \sigma \leq 1$, constant for the family.

Examples

- ⋆ 2D Grids: $\sigma = 1/2 \rightarrow$ 2D Finite Elements;
- ⋆ 3D Grids: $\sigma = 2/3 \rightarrow$ 3D Finite Elements;
- ⋆ Bounded density graphs in dimension d: $\sigma = \frac{d-1}{d}$.

Framework

n^σ-separator theorem

A family of (structurally symmetric) matrices satisfies a n^σ-separator theorem when the graph associated to a matrix can be recursively partitionned as follows:

- ⋆ $G = (V, E)$, $|V| = n$;
- ⋆ $V = A \cup B \cup C$, C topological separator;
- ⋆ $|A|, |B| \leq \alpha n$, $C \leq \beta n^\sigma$;
- ⋆ $0 < \alpha < 1$, $\beta > 0$, $1/2 \leq \sigma \leq 1$, constant for the family.

Examples

- ⋆ 2D Grids: $\sigma = 1/2 \rightarrow$ 2D Finite Elements;
- ⋆ 3D Grids: $\sigma = 2/3 \rightarrow$ 3D Finite Elements;
- ⋆ Bounded density graphs in dimension d: $\sigma = \frac{d-1}{d}$.

Framework

n^σ-separator theorem

A family of (structurally symmetric) matrices satisfies a n^σ-separator theorem when the graph associated to a matrix can be recursively partitionned as follows:

- $G = (V, E)$, $|V| = n$;
- $V = A \cup B \cup C$, C topological separator;
- $|A|, |B| \leq \alpha n$, $C \leq \beta n^\sigma$;
- $0 < \alpha < 1$, $\beta > 0$, $1/2 \leq \sigma \leq 1$, constant for the family.

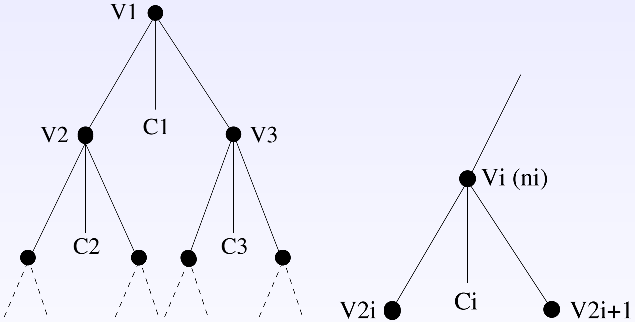
Examples

- 2D Grids: $\sigma = 1/2 \rightarrow$ 2D Finite Elements;
- 3D Grids: $\sigma = 2/3 \rightarrow$ 3D Finite Elements;
- Bounded density graphs in dimension d: $\sigma = \frac{d-1}{d}$.

Partition tree

Partition tree

Recursively partition the graph using such separators

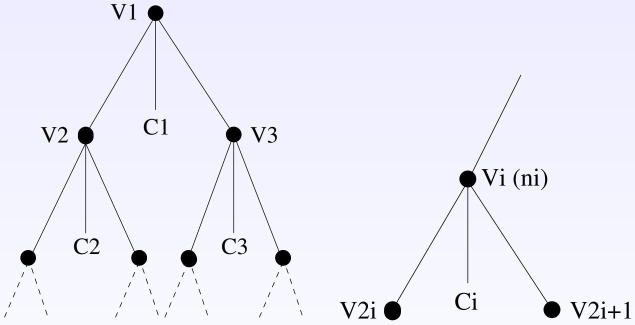


- Binary tree with indices $I_p = [|2^p; 2^{p+1} - 1|]$ at level p $(0 \leq p \leq P)$;
- $(1 - \alpha n_i) - \beta n_i^\sigma \leq n_{2i}, n_{2i+1} \leq \alpha ni$; $|c_i| \leq \beta n_i^\sigma$.

Partition tree

Partition tree

Recursively partition the graph using such separators

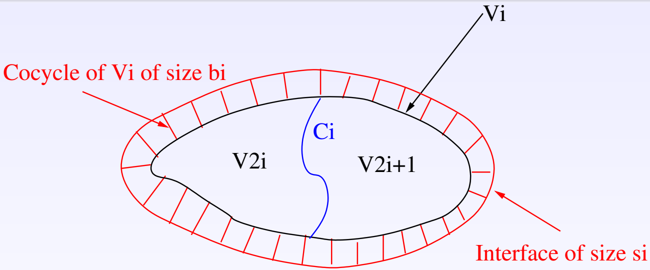


- ⋆ Binary tree with indices $I_p = [|2^p; 2^{p+1} - 1|]$ at level p $(0 \leq p \leq P)$;
- ⋆ $(1 - \alpha n_i) - \beta n_i^\sigma \leq n_{2i}, n_{2i+1} \leq \alpha ni$; $|c_i| \leq \beta n_i^\sigma$.

Partition tree

Partition tree

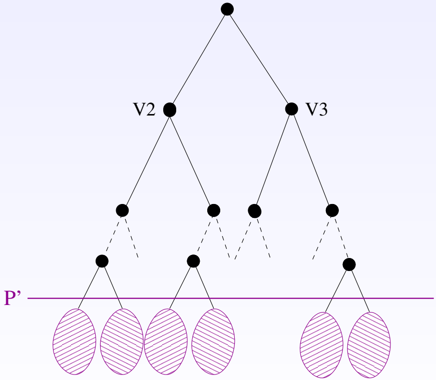
Recursively partition the graph using such separators



Switch point (p')

Switch point

- Top of the tree ($0 \dots p'$ levels): iterative method (Krylov);
- Bottom of the tree: local direct methods.

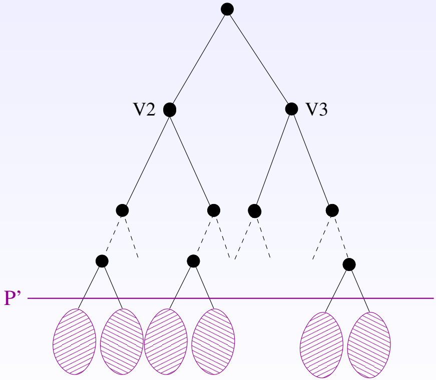


Choose p' such that ...

Switch point (p')

Switch point

- Top of the tree ($0 \dots p'$ levels): iterative method (Krylov);
- Bottom of the tree: local direct methods.



Choose p' such that . . .

Example of application

Diffusion problems

- Upper bound on the number of iterations;
- $\sigma = 2/3$:
  - hybrid: $\theta(n^{4/3})$;
  - direct: $\theta(n^2)$;

Outline

Perspectives

Complexity analysis

- ⋆ Memory requirements;
- ⋆ Study of the parallel case;
- ⋆ Other classes of problems;
- ⋆ Assessing the model with experimental results.

MaPHyS

- ⋆ Integration of other direct solvers (multithreaded PaSTiX);
- ⋆ Integration of other partitioners (Scotch/PT-Scotch);
- ⋆ Compare to other hybrid solvers (Henon et al.; Li et al.).

THANK YOU FOR YOUR ATTENTION

QUESTIONS?