Algebraic domain decomposition solvers: massively parallel scalability study and real life applications

Azzam Haidar (CERFACS) joint work with Luc Giraud (N7-IRIT)

PhD Summary
June 23th, 2008, Toulouse

Outline

Outline

Motivations

$$\mathcal{A}x = b$$

Solution of very Large/Huge *ill*-conditioned sparse linear systems $\mathcal{A}x = b$

- Such problems can require thousands of CPU-hours and many Gigabytes of memory
- Direct solvers:
  - Robust and usually do not fail
  - Memory and computational costs grow nonlinearly
- Iterative solvers:
  - Reduce memory requirements
  - They may fail to converge
  - Typically implemented with preconditioning to accelerate convergence

In an effort to reduce these requirements, a parallel mechanism for combining advantages of those solvers is needed

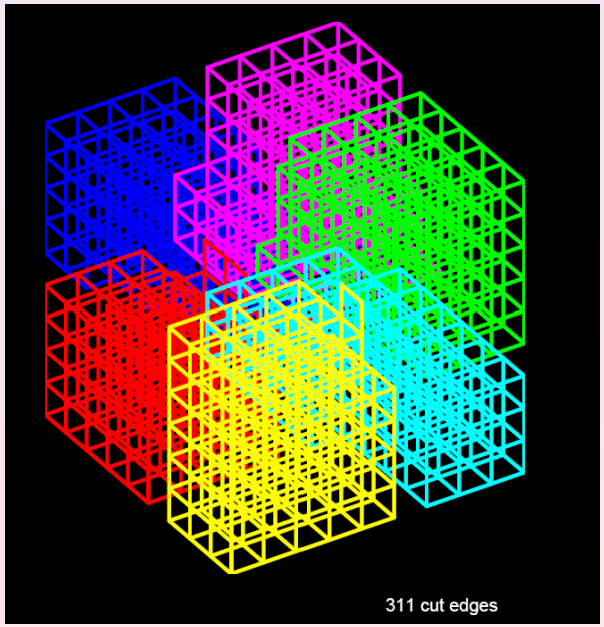
Goal

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)

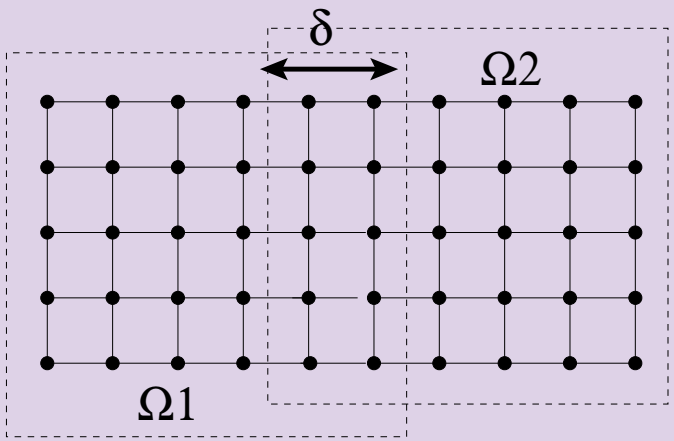
- Natural approach for PDE's
- Extend to general sparse matrices
- Partition the problem into subdomains, subgraphs
- Use a direct solver on the subdomains
- Robust preconditioned iterative solver



Outline

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

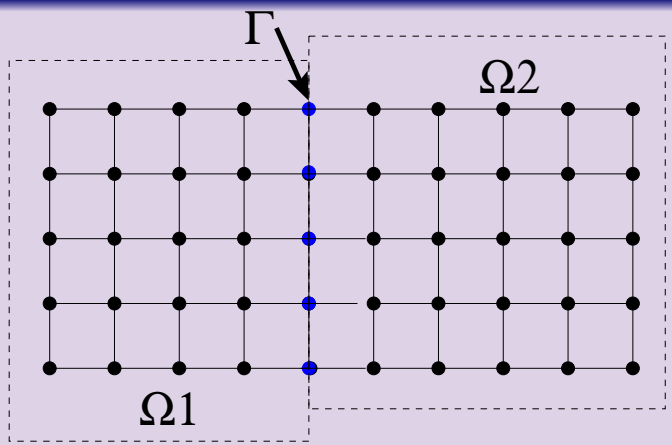
$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{1,1} & \mathcal{A}_{1,\delta} & \\ \mathcal{A}_{\delta,1} & \mathcal{A}_{\delta,\delta} & \mathcal{A}_{\delta,2} \\ & \mathcal{A}_{\delta,2} & \mathcal{A}_{2,2} \end{pmatrix} \Longrightarrow \mathcal{M}^{\delta}_{\mathcal{AS}} = \begin{pmatrix} \boxed{\begin{matrix}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

Classical Additive Schwarz preconditioners N subdomains case

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

Non-overlapping Domain Decomposition

Schur complement reduced system



- *Goal:* solve linear system $\mathcal{A}x = b$
- Apply partially Gaussian elimination
- Solve the reduced system $\mathcal{S}x_\Gamma = f$
- Then solve $\mathcal{A}_i x_i = b_i - \mathcal{A}_{i,\Gamma} x_\Gamma$

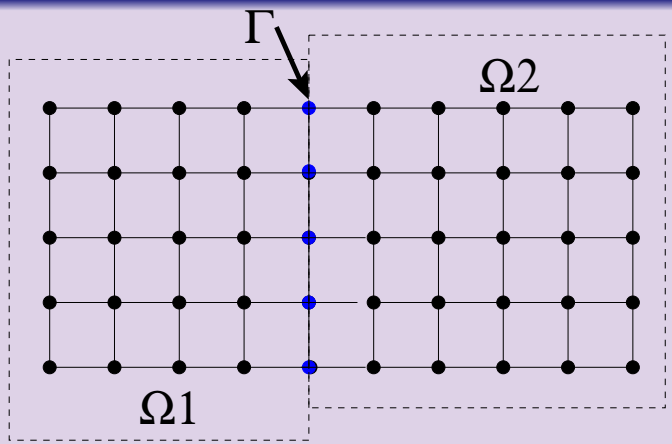
$$\mathcal{A} \qquad x \quad = \quad b \quad \Longrightarrow$$

$$\begin{pmatrix} \mathcal{A}_{1,1} & 0 & \mathcal{A}_{1,\Gamma} \\ 0 & \mathcal{A}_{2,2} & \mathcal{A}_{2,\Gamma} \\ \mathcal{A}_{\Gamma,1} & \mathcal{A}_{\Gamma,2} & \mathcal{A}_{\Gamma,\Gamma} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_\Gamma \end{pmatrix} \Longrightarrow$$

$$\begin{pmatrix} \mathcal{I} & 0 & 0 \\ 0 & \mathcal{I} & 0 \\ \mathcal{A}_{\Gamma,1}\mathcal{A}_{1,1}^{-1} & \mathcal{A}_{\Gamma,2}\mathcal{A}_{2,2}^{-1} & \mathcal{I} \end{pmatrix} \begin{pmatrix} \mathcal{A}_{1,1} & 0 & \mathcal{A}_{1,\Gamma} \\ 0 & \mathcal{A}_{2,2} & \mathcal{A}_{2,\Gamma} \\ 0 & 0 & \mathcal{A}_{\Gamma,\Gamma} - \sum_{i=1}^{2} \mathcal{A}_{\Gamma,i}\mathcal{A}_{i,i}^{-1}\mathcal{A}_{i,\Gamma} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_\Gamma \end{pmatrix}$$

Non-overlapping Domain Decomposition

Schur complement reduced system



- *Goal:* solve linear system $\mathcal{A}x = b$
- Apply partially Gaussian elimination
- Solve the reduced system $\mathcal{S}x_\Gamma = f$
- Then solve $\mathcal{A}_i x_i = b_i - \mathcal{A}_{i,\Gamma} x_\Gamma$

$$\begin{pmatrix} \mathcal{A}_{1,1} & 0 & \mathcal{A}_{1,\Gamma} \\ 0 & \mathcal{A}_{2,2} & \mathcal{A}_{2,\Gamma} \\ 0 & 0 & \mathcal{S} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_\Gamma - \sum_{i=1}^{2} \mathcal{A}_{\Gamma,i}\mathcal{A}_{i,i}^{-1} b_i \end{pmatrix}$$

Solve $\mathcal{A}x = b \Longrightarrow$ solve the reduced system $\mathcal{S}x_\Gamma = f \Longrightarrow$ then solve $\mathcal{A}_i x_i = b_i - \mathcal{A}_{i,\Gamma} x_\Gamma$

where $\mathcal{S} = \mathcal{A}_{\Gamma,\Gamma} - \sum_{i=1}^{2} \mathcal{A}_{\Gamma,i}\mathcal{A}_{i,i}^{-1}\mathcal{A}_{i,\Gamma}$,

and $f = b_\Gamma - \sum_{i=1}^{2} \mathcal{A}_{\Gamma,i}\mathcal{A}_{i,i}^{-1} b_i$.

Parallel implementation for solving $\mathcal{A}x = b$

- 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"

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

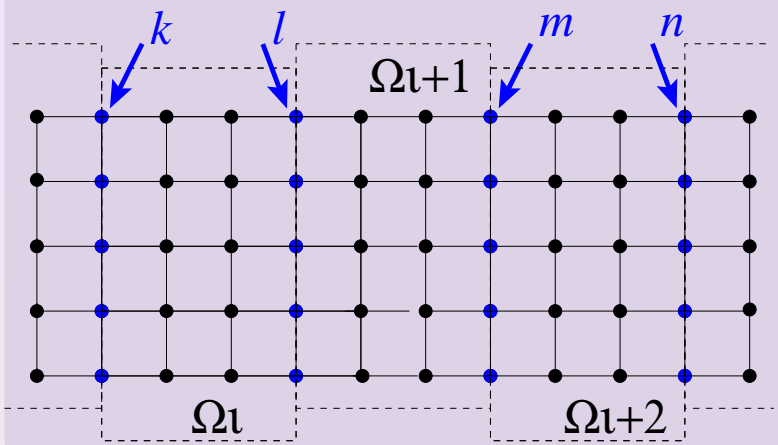
- The reduced system $\mathcal{S}x_\Gamma = f$ is solved using a distributed Krylov solver
  - One matrix vector product per iteration each processor computes $\mathcal{S}^{(i)}(x^{(i)}_\Gamma)^k = (y^{(i)})^k$
  - One local preconditioner apply $(\mathcal{M}^{(i)})(z^{(i)})^k = (r^{(i)})^k$
  - Local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

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

Outline

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.Giraud, G.Meurant - 01]

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

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

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

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

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

$$\bar{\mathcal{S}}^{(i)} = \mathcal{S}^{(i)} + \sum_{\iota \in adj} \mathcal{S}_{\ell\ell}^{(\iota)}$$

Algebraic Additive Schwarz preconditioner

Main characteristics in 2D [PhD of J. C. Rioual - 02]

- The ratio interface/interior is small
- Does not require large amount of memory to store the preconditioner
- Computation/application of the preconditioner are fast
- They consist in a call to LAPACK/BLAS-2 kernels

Main characteristics in 3D

- The ratio interface/interior is large
- The storage of the preconditioner might not be affordable
- The computation/application cost of the preconditioner might penalize the method
- Need cheaper Algebraic Additive Schwarz form of the preconditioner

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

- 2048 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

Local direct solver : MUMPS [P.Amestoy, I.Duff, J.Koster, J.Y.L'Excellent - 01]

- Main features
  - Parallel distributed multifrontal solver (F90, MPI)
  - Symmetric and Unsymmetric factorizations
  - Element entry matrices, distributed matrices
  - Efficient Schur complement calculation
  - Iterative refinement and backward error analysis
- Public domain: new version 4.7.3
  `http://mumps.enseeiht.fr/`

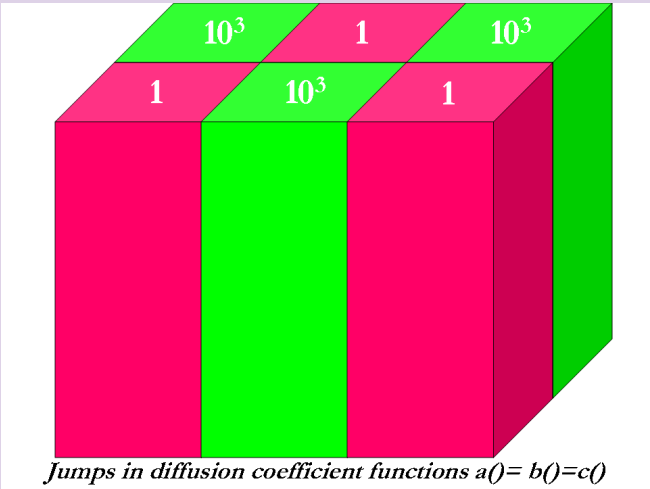
Iterative solver : KRYLOV

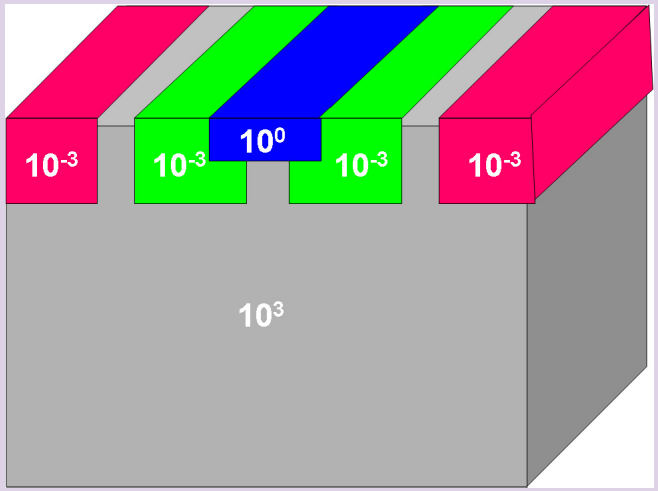
- Symmetric positive definite
  - Parallel distributed Conjugate gradient solver [V.Frayssé, L.Giraud - 00]
- Unymmetric or undefinite symmetric
  - Parallel distributed GMRES/FGMRES solver [V.Frayssé, L.Giraud, S.Gratton - 97]
- Public domain:
  `http://www.cerfacs.fr/algor/Softs/`

Outline

Academic model problems

Problem patterns





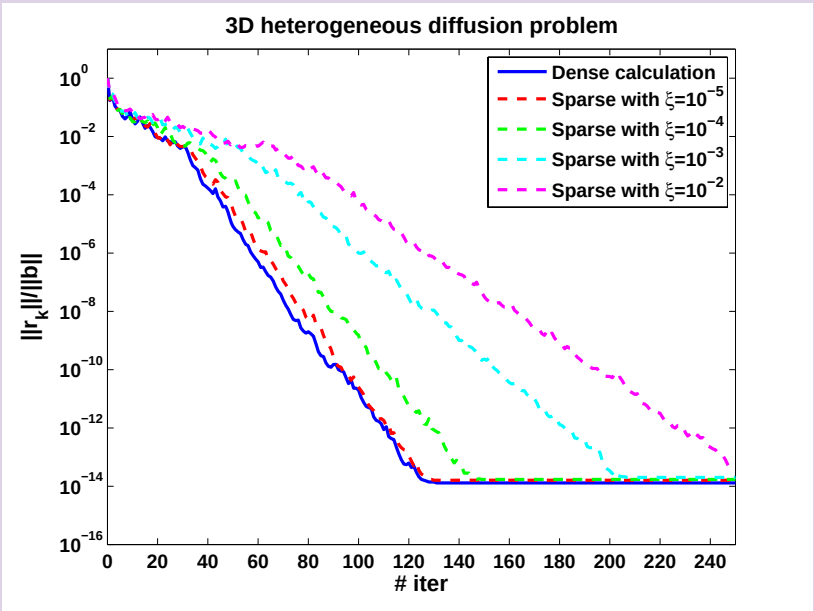
Diffusion equation ($\epsilon = 1$ and $v = 0$)

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

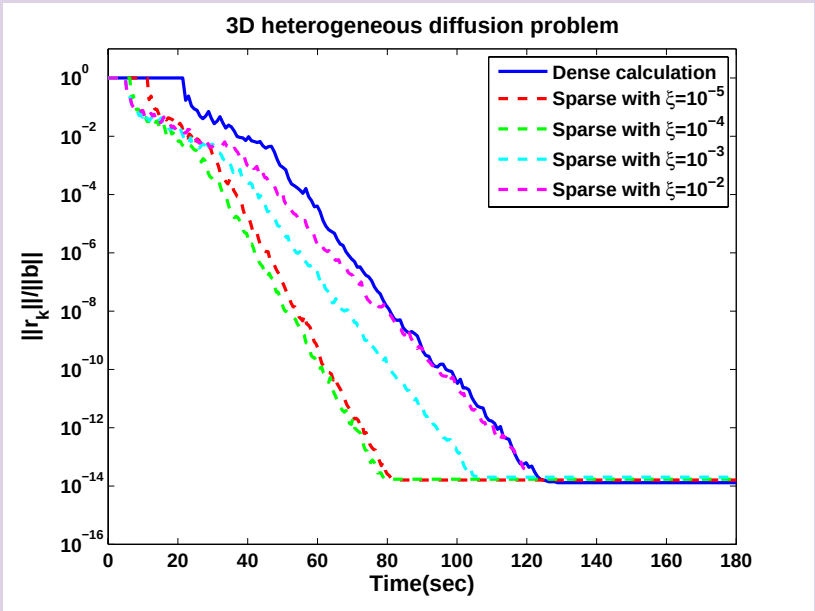
- Classical Poisson problems
- Heterogeneous problems
- Anisotropic-heterogeneous problems
- Convection dominated term

Numerical behaviour of sparse preconditioners

Convergence history of PCG



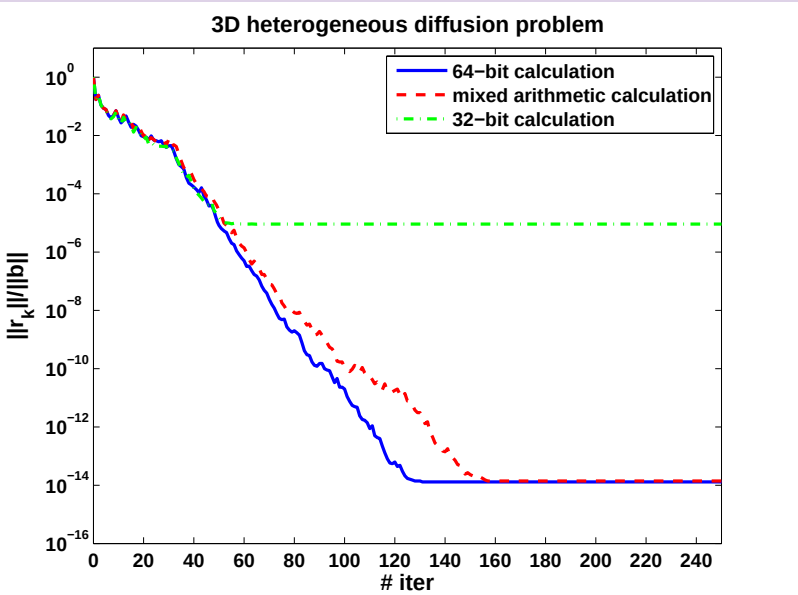
Time history of PCG



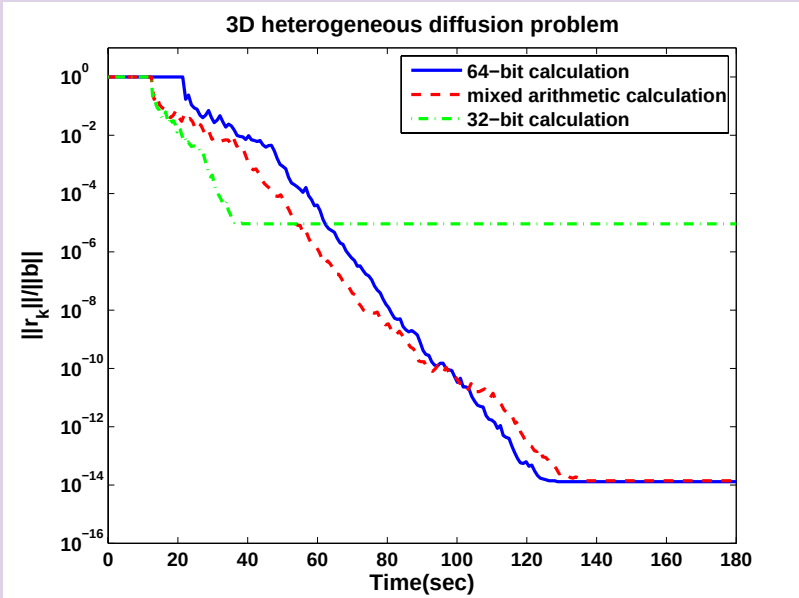
- 3*D* heterogeneous diffusion problem with 43 Mdof mapped on 1000 processors
- For (ξ $\lll$) the convergence is marginally affected while the memory saving is significant 15%
- For (ξ $\ggg$) a lot of resources are saved but the convergence becomes very poor 1%
- Even though they require more iterations,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

Numerical behaviour of mixed preconditioners

Convergence history of PCG



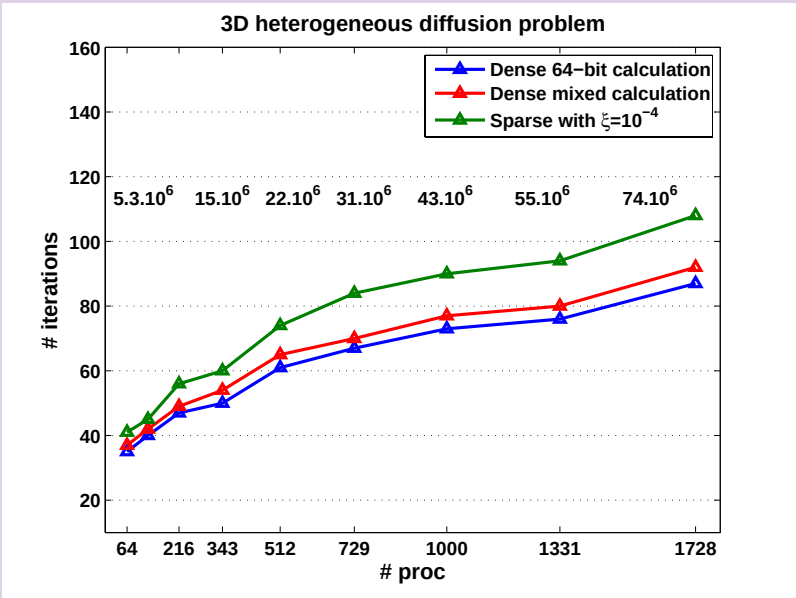
Time history of PCG



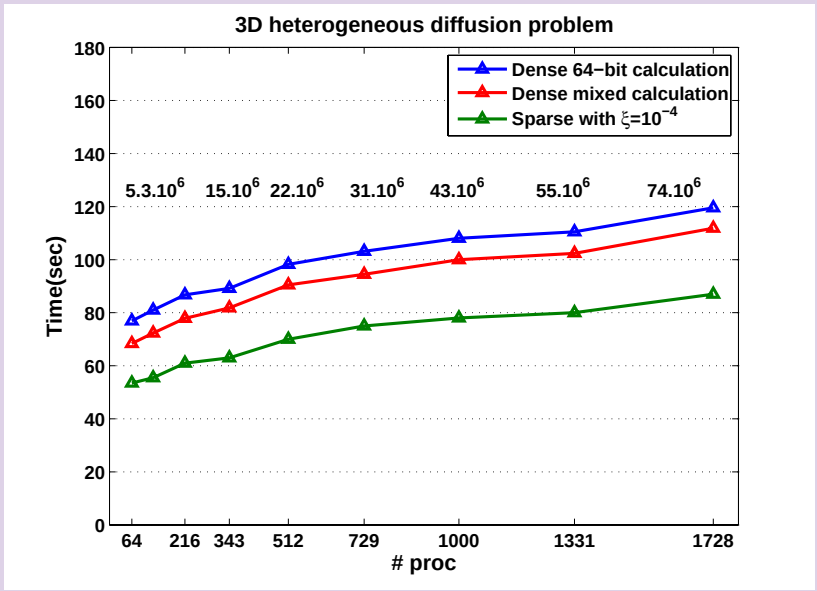
- 3D heterogeneous diffusion problem with 43 Mdof mapped on 1000 processors
- 64-bit and mixed computation both attained an accuracy at the level of 64-bit machine precision
- The number of iterations slightly increases
- The mixed approach is the fastest, down to an accuracy that is problem dependent

Scaled scalability on massively parallel platforms

Numerical scalability



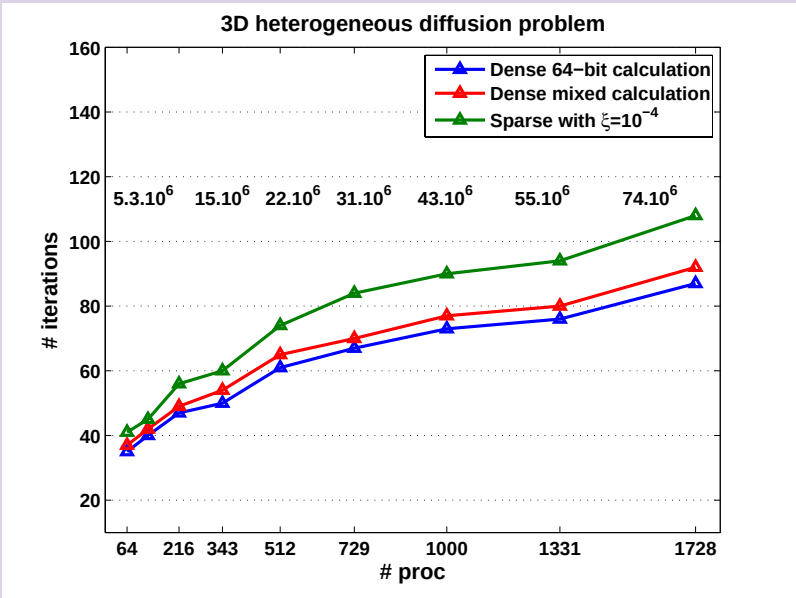
Parallel performance



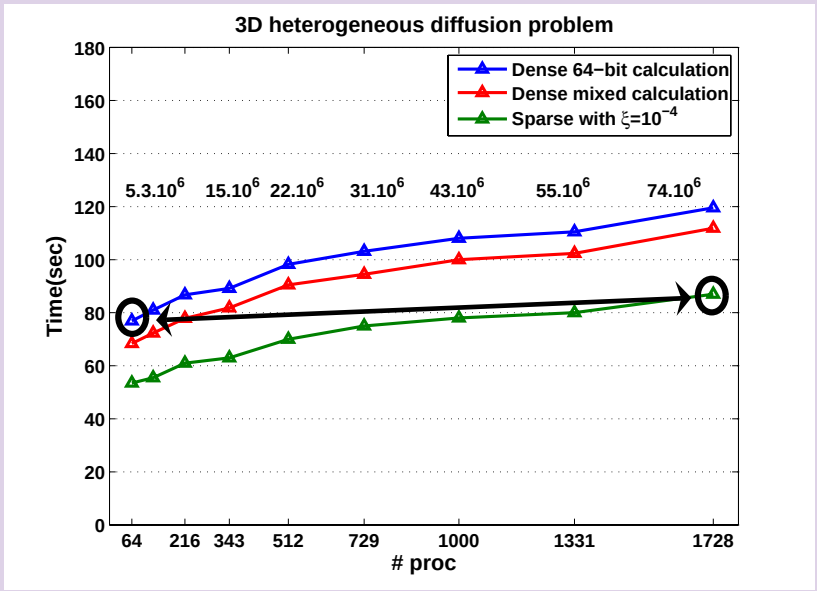
- The solved problem size varies from 2.7 up to 74 Mdof
- Control the grow in the # of iterations by introducing a coarse space correction
- The computing time increases slightly when increasing # sub-domains
- Although the preconditioners do not scale perfectly, the parallel time scalability is acceptable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

Scaled scalability on massively parallel platforms

Numerical scalability



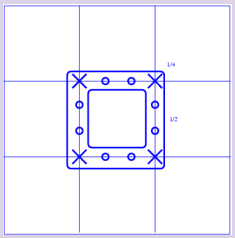
Parallel performance



- 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

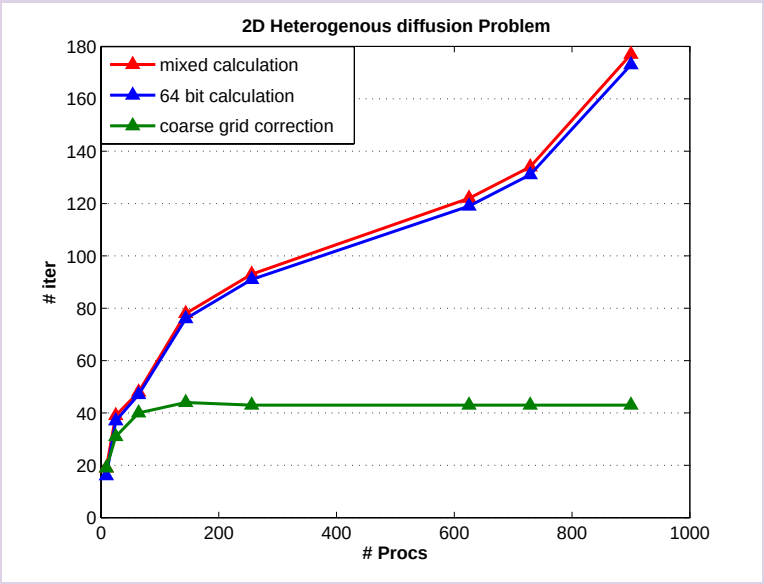
Numerical alternative: numerical scalability in 3D

Domain based coarse space : $M = M_{AS} + R_O^T A_O^{-1} R_0$ where $A_0 = R_0 S R_O^T$

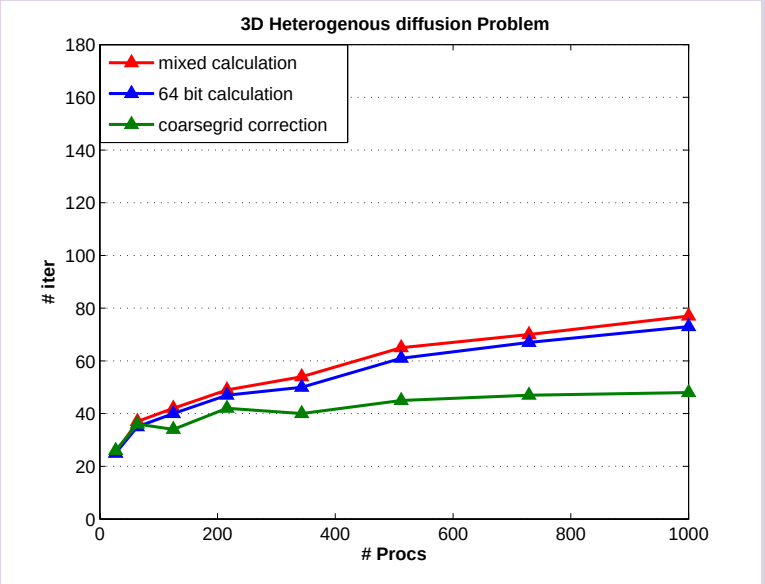


- "As many" dof in the coarse space as sub-domains [Carvalho, Giraud, Le Tallec, 01]
- Partition of unity : R_0^T simplest constant interpolation

2D Heterogenous diffusion

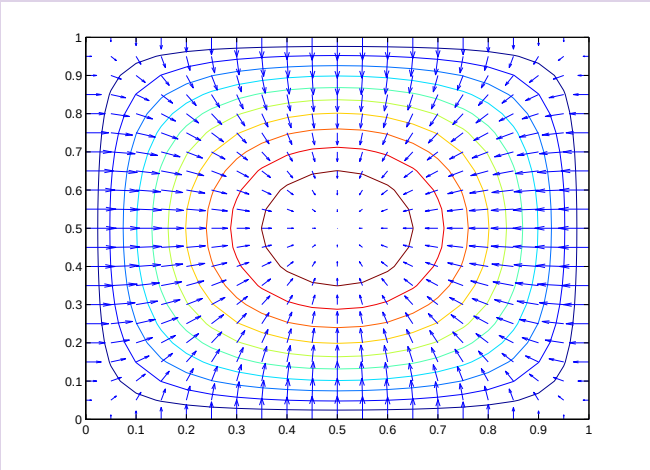


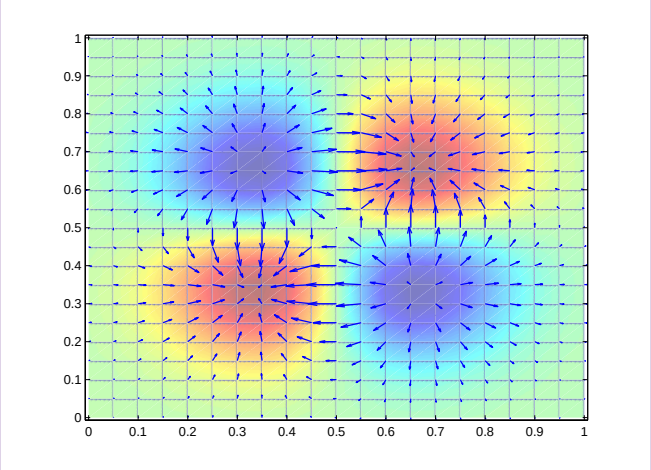
3D Heterogenous diffusion



Academic model problems

Problem patterns





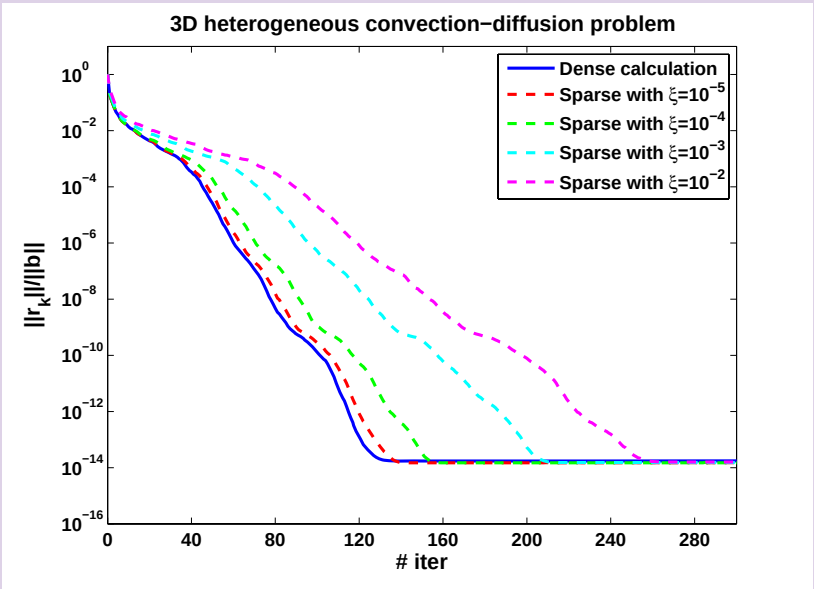
Convection-diffusion equation

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

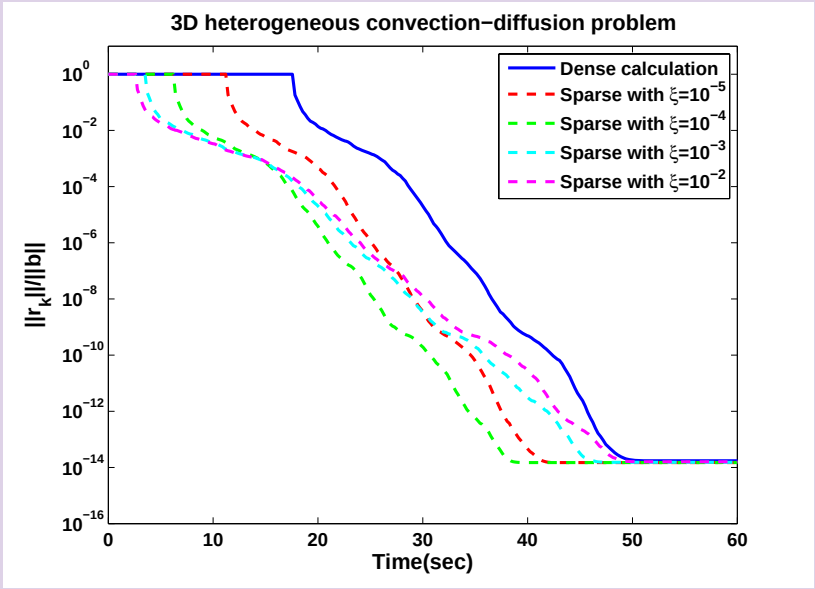
- Classical Poisson problems
- Heterogeneous problems
- Anisotropic-heterogeneous problems

Numerical behaviour of sparse preconditioners

Convergence history of GMRES



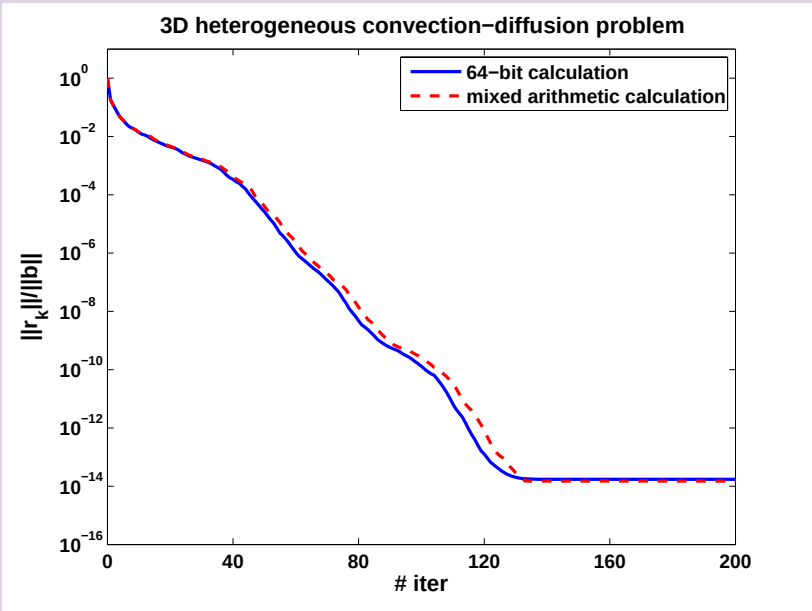
Time history of GMRES



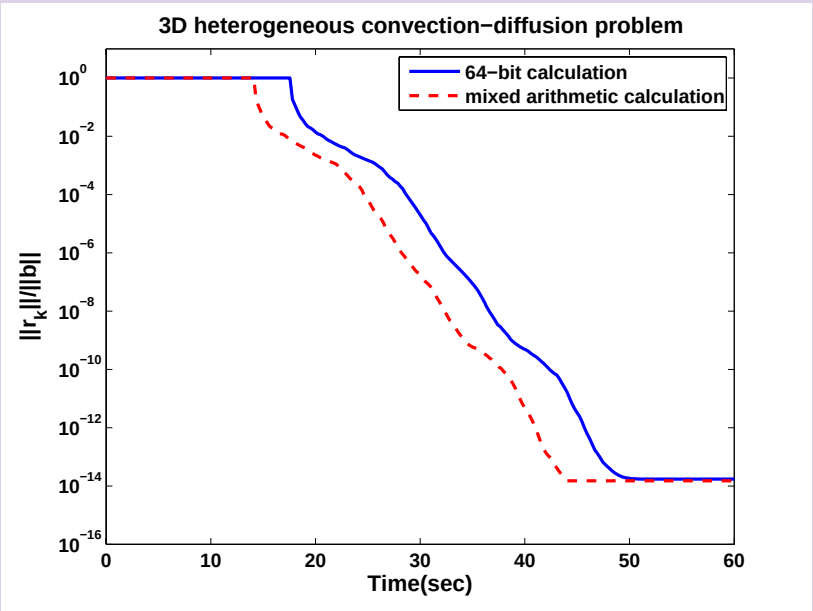
- 3D heterogeneous convection-diffusion problem of 27 Mdof mapped on 1000 processors
- For (ξ $\lll$) the convergence is marginally affected while the memory saving is significant
- For (ξ $\ggg$) a lot of resources are saved but the convergence becomes very poor
- Even though they require more iterations, the sparsified variants converge faster as the time per iteration is smaller and the setup of the preconditioner is cheaper.

Numerical behaviour of mixed preconditioners

Convergence history of FGMRES



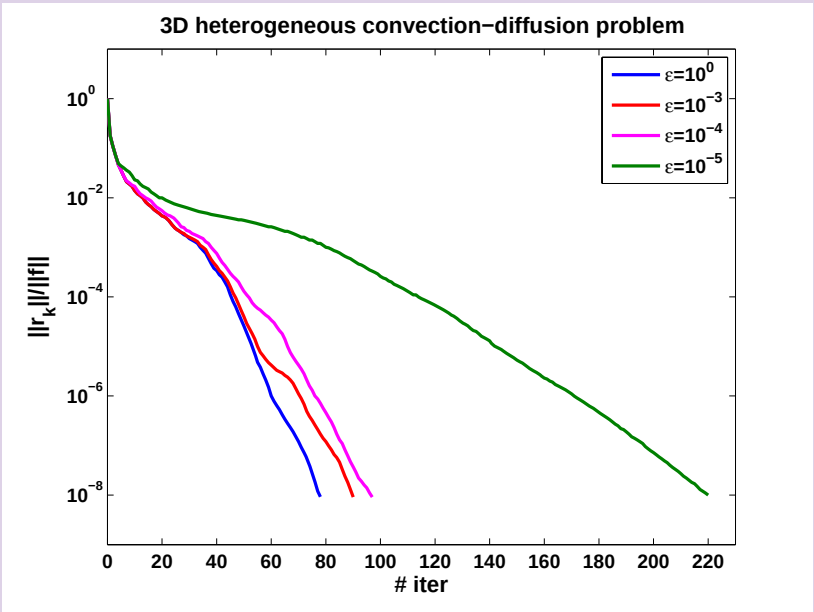
Time history of FGMRES



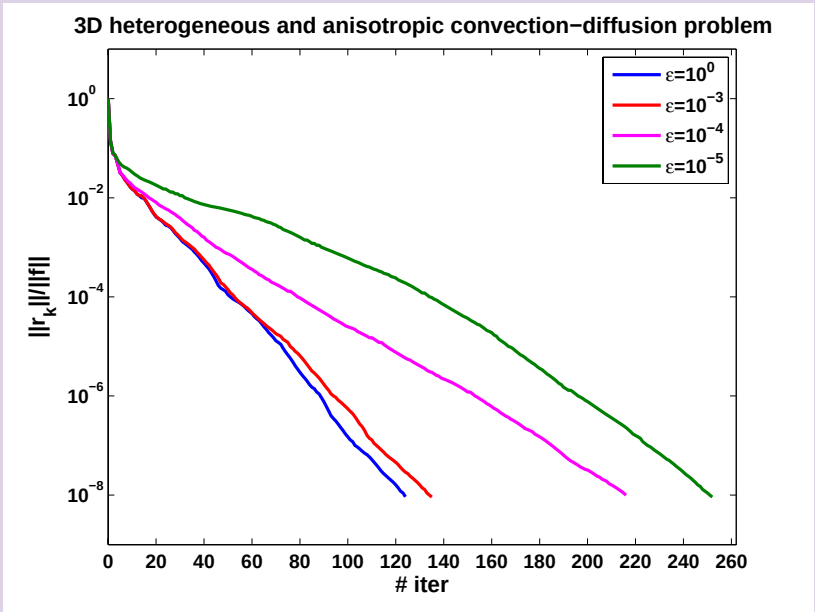
- 3D heterogeneous convection-diffusion problem of 27 Mdof mapped on 1000 processors
- 64-bit and mixed computation both attained an accuracy at the level of 64-bit machine precision using the FGMRES solver
- The mixed arithmetic implementation compares favorably with the 64-bit one.
- The saving in computing time of the mixed approach is less distinctive due the platform

Effect of the Péclet number

Effect of the convection term



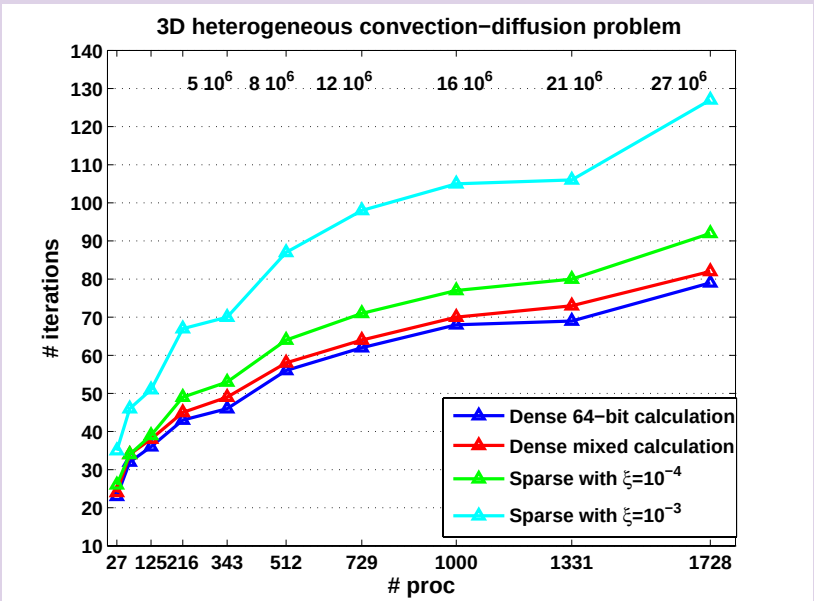
Effect of the convection term



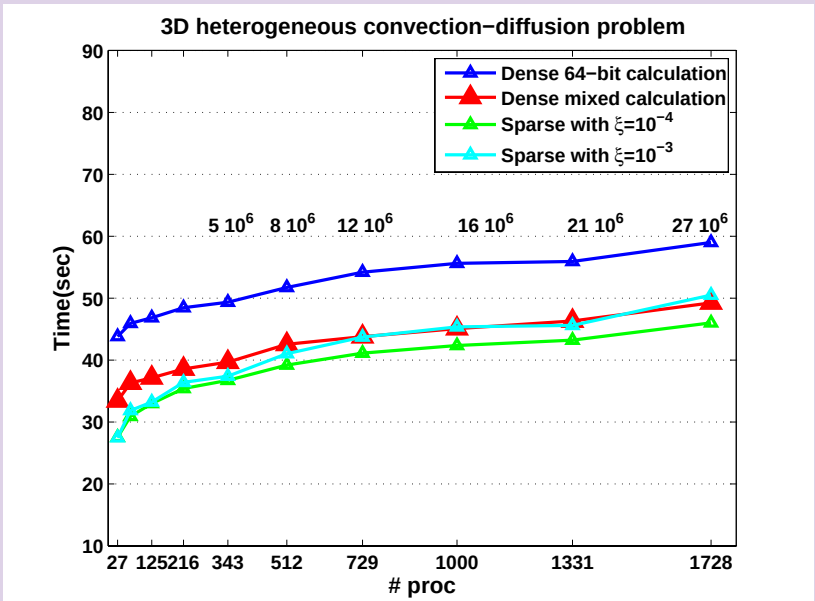
- 3D convection-diffusion problems of 27 Mdof dof mapped on 1000 processors
- Increasing the convection term makes harder the problem to solve

Scaled scalability on massively parallel platforms

Numerical scalability



Parallel performance



- The mixed preconditioner performs very similarly
- The sparser the preconditioner, the larger gap in the number of iterations is
- Even if the number of iterations to converge increases as the number of subdomains increases, the parallel scalability of the preconditioners remains acceptable
- The trend is similar for all variants of the preconditioner using GMRES/FGMRES solver

Summary on the model problems [L.Giraud, A.Haidar, L.T.Watson - 08]

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

Mixed preconditioner

- Mixed arithmetic and 64-bit both attained an accuracy at the level of 64-bit machine precision
- Mixed preconditioner does not delay that much the convergence

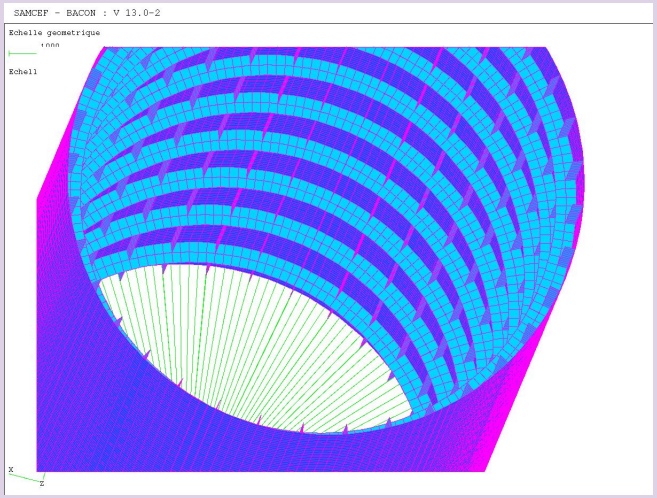
On the parallel scalability

- 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

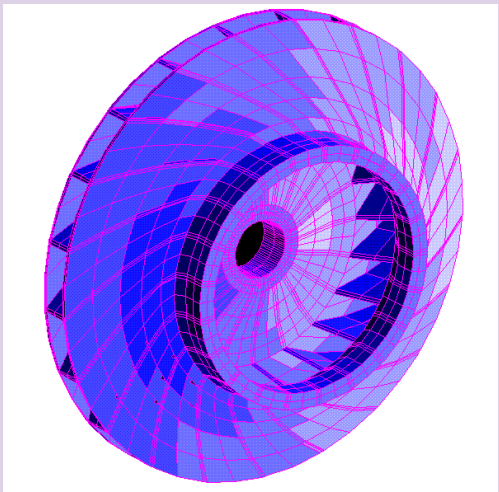
Indefinite systems in structural mechanics S.Pralet, SAMTECH

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

Partitioning strategies

Main characteristics

- Linear elasticity equations with constraints such as rigid bodies and cyclic conditions
- Lagrange multipliers $\Longrightarrow$ symmetric indefinite augmented systems

Numerical difficulties

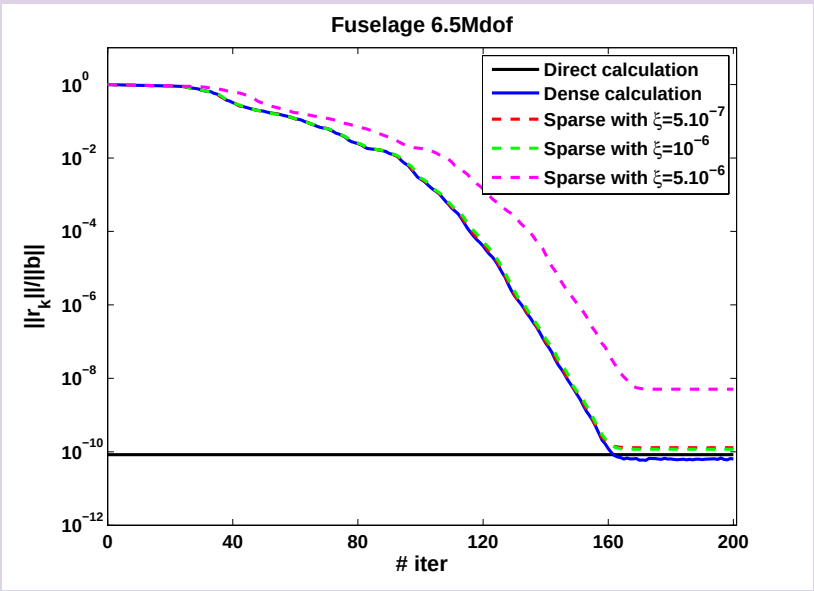
- The local matrix associated with the internal unknowns might be structurally singular
- Fix Lagrange multipliers difficulties
- Idea: enforce the Lagrange multipliers to be moved into the interface

Performance difficulties

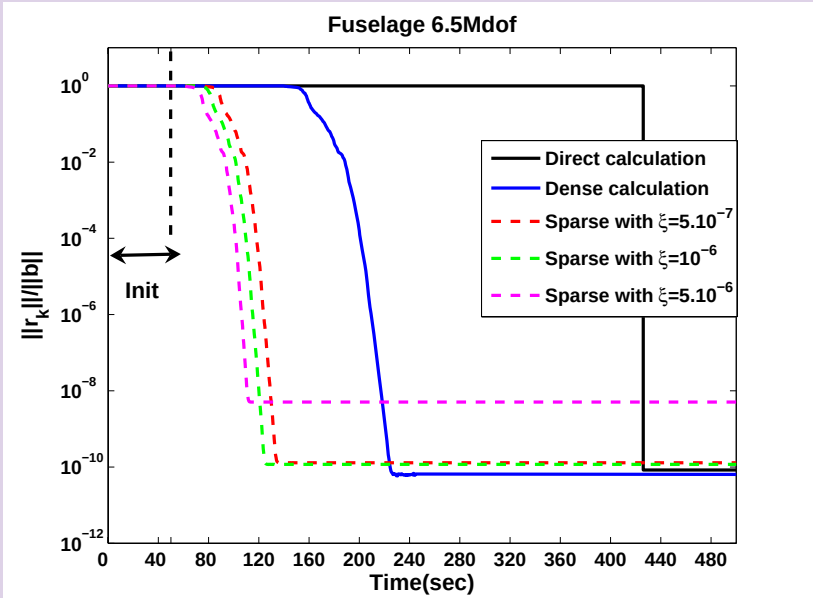
- Needs to balance and optimize the distribution of the Lagrange multipliers among the balanced subdomains
- Apply constraint (weights) to the partitioner (dual mesh graph)

Numerical behaviour of sparse preconditioners

Convergence history



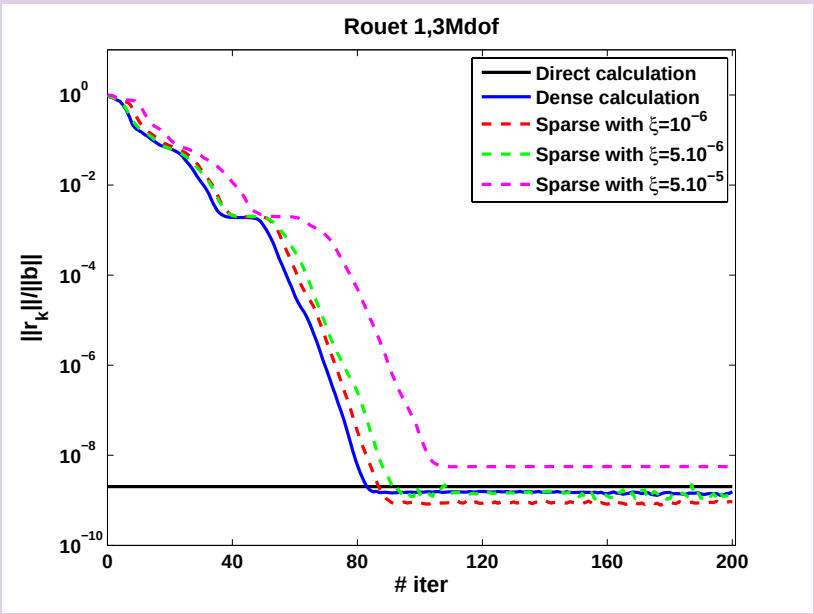
Time history



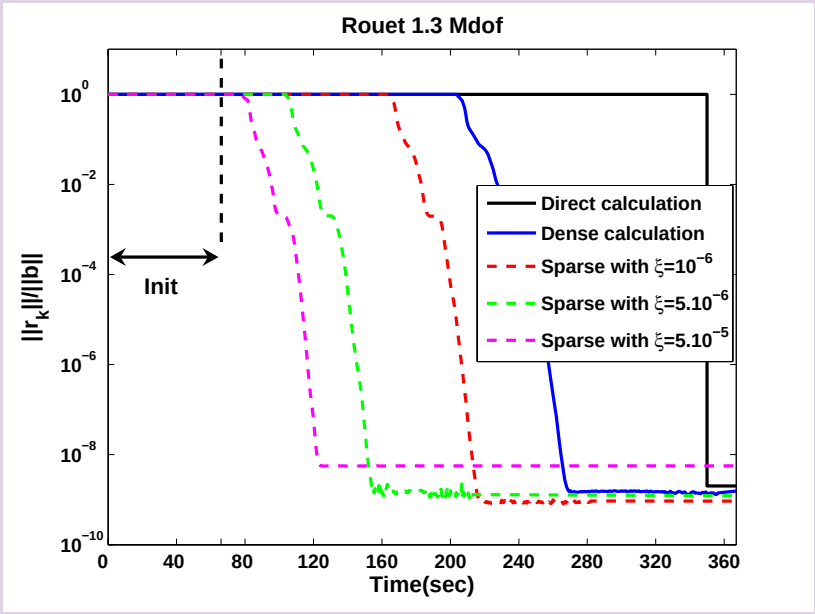
- Fuselage problem of 6.5 Mdof dof mapped on 16 processors
- The sparse preconditioner setup is 4 times faster than the dense one (19.5 v.s. 89 seconds)
- In term of global computing time, the sparse algorithm is about twice faster
- The attainable accuracy of the hybrid solver is comparable to the one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

Numerical behaviour on sparse preconditioners

Convergence history



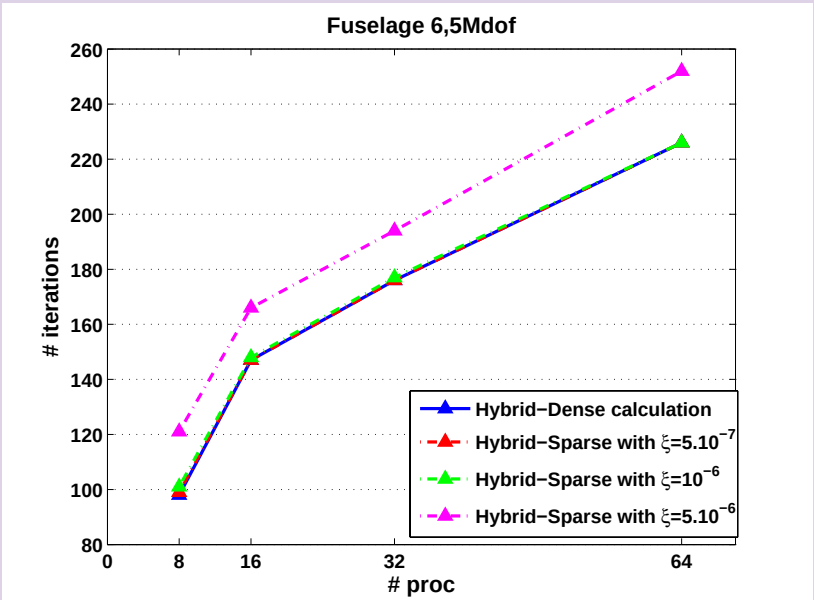
Time history



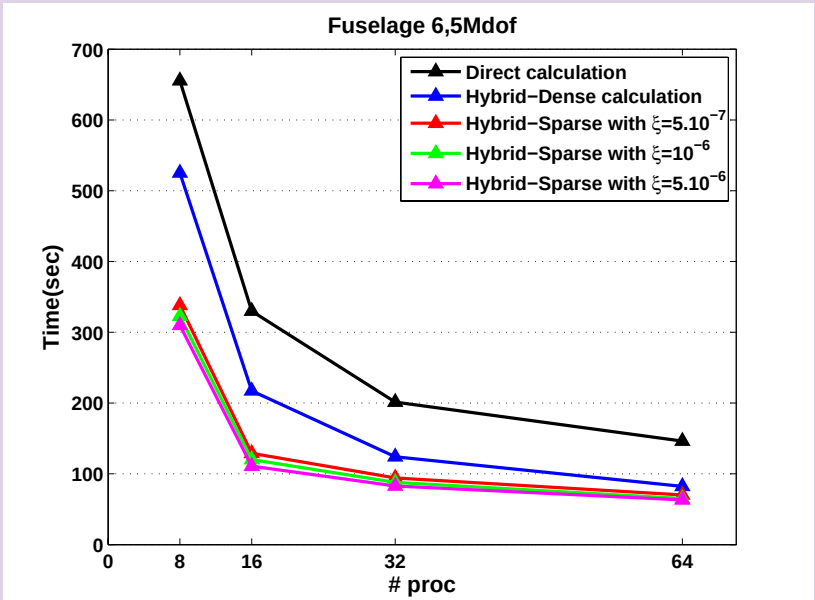
- Rouet problem of 1.3Mdof dof mapped on 16 processors
- The sparsified variant outperforms its dense counterpart
- The hybrid techniques solution is comparable to the one obtain with the direct solver

Performance on indefinite systems

Numerical scalability



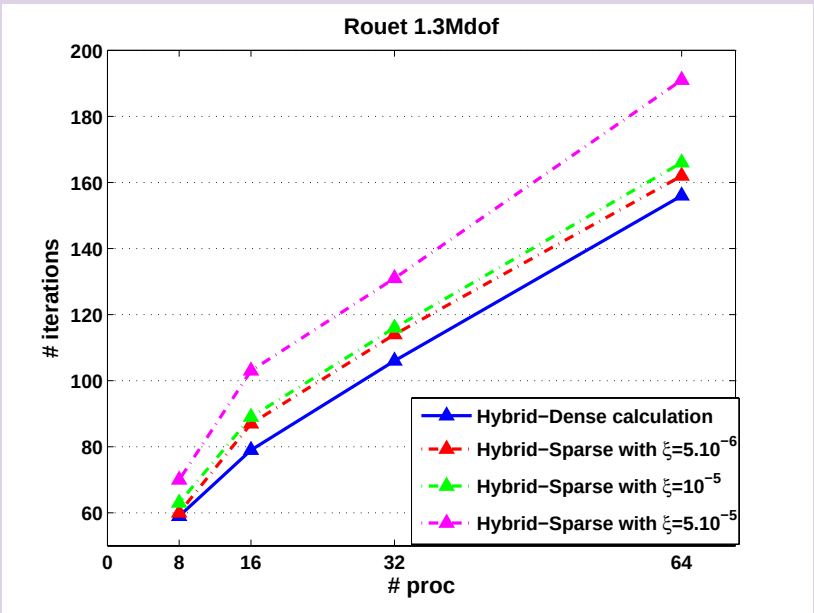
Parallel performance



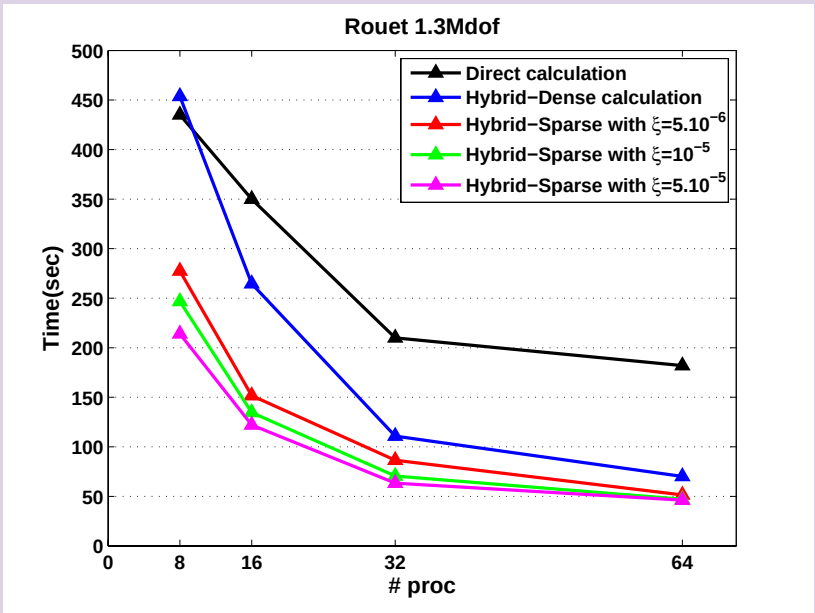
- Fixed problem size: increasing the # of subdomains $\Longrightarrow$ an increase in the # of iterations
- Attractive speedups can be observed
- The sparsified variant the most efficient (CPU, memory)

Performance on indefinite systems

Numerical scalability



Parallel performance



- Fixed problem size: increasing the # of subdomains $\Longrightarrow$ a linear (slight??) growth in the # of iterations
- Very attractive speedups can be observed
- The sparsified variant is of great interest
- Compared with the sparse direct solver, the hybrid approach gives always the fastest scheme

Summary on the structural mechanics problems

Characteristics of the hybrid approach

- It was observed that the mixed precision algorithm behaves very closely to the 64-bit algorithm
- The sparse preconditioner represents a significant saving in computing ressources
- Relax the stopping criterion when embeded into a nonlinear solver

Difficulties

- The number of iterations increases when increasing the number of subdomains
- Large amount of memory storage is required for such engineering applications

One of the solutions

- Exploit *2-levels* of parallelism

Exploiting *2-levels* of parallelism - motivations

"*The numerical improvement*"

- Classical parallel implementations (*1-level*) of DD assign one subdomain per processor
- Parallelizing means increasing the number of subdomains
- Increasing the number of subdomains often leads to increasing the number of iterations
- To avoid this, one can instead of increasing the number of subdomains, keeping it small while handling each subdomain by more than one processor introducing *2-levels* of parallelism

"*The parallel performance improvement*"

- Large 3*D* systems often require a huge amount of data storage
- On SMP node: classical *1-level parallel* can only use a subset of the available processors
- Thus some processors are "wasted", as they are "idle" during the computation
- The "idle" processors might contribute to the computation and the simulation runs closer to the peak of per-node performance by using *2-levels* of parallelism

Numerical improvement benefits

Fuselage of 6.5Mdof

# total processors	Algo	# subdomains	# processors/ subdomain	# iter	iterative loop time
16 processors	*1-level parallel*	16	1	147	77.9
	2-level parallel	8	2	98	51.4
32 processors	*1-level parallel*	32	1	176	58.1
	2-level parallel	16	2	147	44.8
	2-level parallel	8	4	98	32.5
64 processors	*1-level parallel*	64	1	226	54.2
	2-level parallel	32	2	176	40.1
	2-level parallel	16	4	147	31.3
	2-level parallel	8	8	98	27.4

- Reduce the number of subdomains $\Longrightarrow$ reduce the number of iterations
- Though the subdomain size increases, the time of the iterative loop decreases as:
  - The number of iterations decreases
  - Each subdomain is handled in parallel
  - All the iterative kernels are efficiently computed in parallel
- The speedup factors of the iterative loop vary from 1.3 to 1.8
- Very attractive especially when the convergence rate depends on the # of subdomains

Parallel performance benefits

Fuselage of 6.5Mdof

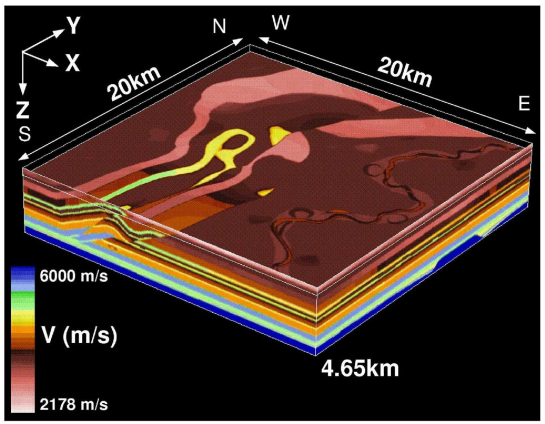
# subdomains or SMP node	Algo	proc/subdom or "*working*"	Precond setup time	# iter	iterative loop time	Total time
8	*1-level*	1	208.0	98	94.1	525.1
	2-level	2	124.6		51.5	399.1
		4	70.8		32.5	326.4
16	*1-level*	1	89.0	147	77.9	217.2
	2-level	2	52.7		44.8	147.8
		4	30.4		31.3	112.0
32	*1-level*	1	30.0	176	58.1	124.1
	2-level	2	20.4		40.8	97.2
		4	13.0		22.7	71.7

- When running large simulations that need all the memory available on the nodes
- The *1-level parallel* algo "wastes" ressource performance (it lose 48 "idle" processors on 16 SMP)
- The *2-level parallel* algo exploits the computing facilities of the remaining "idle" processors
- The *2-level parallel* algo runs closer to the peak of per-node performance
- The preconditioner setup time benefits vary from 1.5 to 3
- The speedup factors of the iterative loop vary from 1.8 to 2.7

Outline

Seismic modelling applications SEISCOPE consortium

3D SEG/EAGE Overthrust model

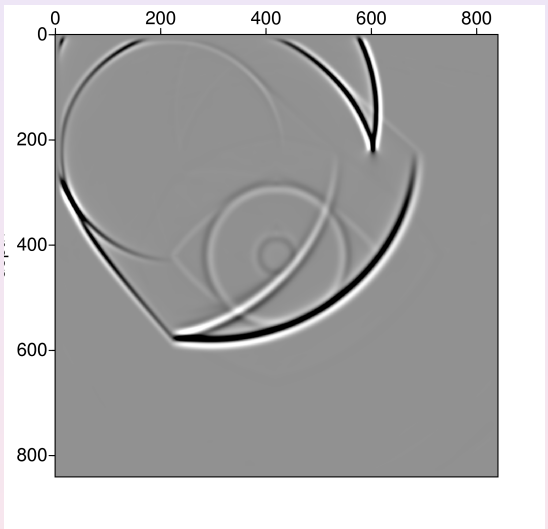


- Solve the Helmholtz equation
- $20 \times 20 \times 4.65\, km^3$
- 4 grid points per minimum wavelength
- PML (Perfectly-Matched Layer) [J.P.Berenger - 94]
- 5.6 Mdof at 7 Hz

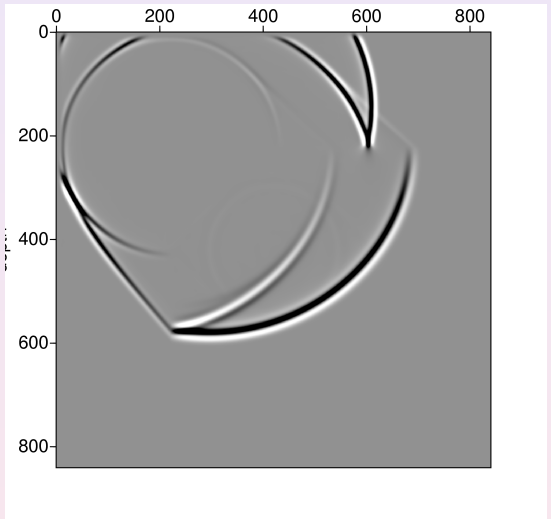
Main characteristics

- Frequency-domain full-waveform tomography [F.Sourbier, S.Operto, J.Virieux - 08]
- The inversion of a few frequencies are enough to build velocity models
- Multisource frequency-domain wave modeling requires the solution of multiple RHS
- Traditional method of choice for solving these systems relies on sparse direct solvers
- To overcome this limitation, the goal is to develop efficient hybrid methods for large 3D

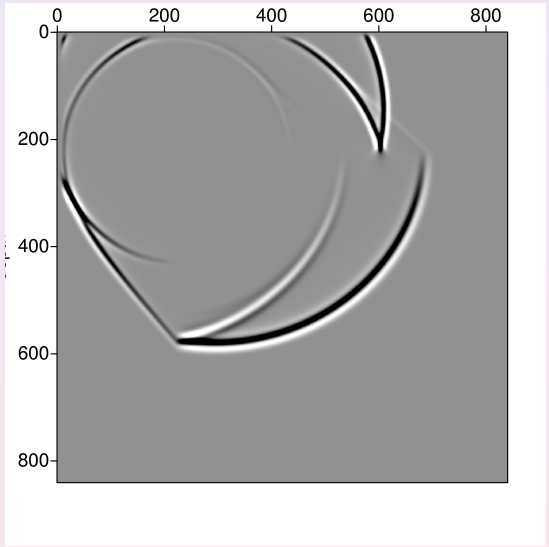
Comments on the stopping criterion



$\eta_b = 10^{-1}$.



$\eta_b = 10^{-2}$.



$\eta_b = 10^{-3}$.

Hybrid solver for 2×2 subdomains on the 2D corner-edge model.

Performance on unsymmetric complex systems

3D SEG/EAGE Overthrust model at 7Hz

subdomains			Memory	Initial-ization	Precond-itioner	# of iterations	Time per RHS	Total time
#	size	interface	All (GB)					
50	$54 \times 54 \times 31$	11056	191.6	614	497.1	81	67.8	1178.9
72	$45 \times 45 \times 31$	8833	179.3	334	273.5	103	73.9	681.4
96	$45 \times 33 \times 31$	7405	167.8	184	153.8	119	61.1	398.9
98	$38 \times 38 \times 31$	7216	169.7	189	141.5	148	66.7	397.2
192	$33 \times 33 \times 21$	5578	147.4	90	78.2	235	85.8	254.0

- Complex shifted variant of the Algebraic Additive Schwarz preconditioner is used
- Increasing the number of subdomains reduces the memory requirement for hybrid solver
- Increasing the number of subdomains reduces the memory requirement for direct solver

Numerical improvement benefits

2-levels of parallelism on 3*D* Overthrust SEG/EAGE

Frequency equal to 7 *Hz*						
Available processors	Algo	# subdomains	Processors/ subdomain	# iter	Iterative loop	Time per RHS
≋ 200 processors	*1-level parallel*	192	1	235	79.0	85.8
	2-level parallel	96	2	119	38.2	45.1
	2-level parallel	50	4	81	28.1	35.5
≋ 100 processors	*1-level parallel*	96	1	119	57.0	61.1
	1-level parallel	98	1	148	66.7	66.7
	2-level parallel	50	2	81	39.1	45.1

- Reduce the number of subdomains $\Longrightarrow$ reduce the number of iterations
- Though the subdomain size increases, the time of the iterative loop decreases as:
- The speedup factor of one RHS simulation vary from 1.3 to 2.5
- Very attractive approach for multiple right-hand sides simulations

Parallel performance benefits

2-levels of parallelism on 3D Overthrust SEG/EAGE

# subdomains	Algo	proc/subdom or "*working*"	Precond setup time	# iter	iterative loop time	Time per RHS	Total time
50	*1-level*	1	497.1	81	64.4	67.8	1178.9
	2-level	2	262.4		39.1	45.1	854.5
		4	135.3		28.1	35.5	419.8
81	*1-level*	1	256.3	109	73.6	77.4	557.7
	2-level	2	169.2		53.7	57.2	431.4
96	*1-level*	1	153.8	119	57.0	61.1	398.9
	2-level	2	81.2		38.2	45.1	299.3

- The preconditioner setup time benefits vary from 1.5 to 3.6
- One RHS simulation time is improved by a factor of 1.3 to 1.7

Outline

Perspectives

PHyLeaS project International INRIA "Associate Team"

- Reduce the preconditioner cost
- Study alternative strategies to get sparse approximation of the local Schur
- Special attention would have to be paid to ensure a good work balance strategy

Solstice project ANR-06-CIS6- 010

- Study the behavior of the preconditioner on more general problems
- Extend this algebraic techniques to general sparse matrices

pILU-based sparse approximation joint work with Y. Saad

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

3*D* Poisson problem (# PCG iterations)

		# subdomains ≡ # processors							
subdomain grid size		27	64	125	216	343	512	729	1000
$20 \times 20 \times 20$	M_{d-64}	16	23	25	29	32	35	39	42
	M_{sp-64}	16	23	26	31	34	39	43	46
	$M_{\phi LeaS}$	22	29	32	39	43	48	52	58

3*D* heterogeneous diffusion (# PCG iterations)

		# subdomains ≡ # processors							
subdomain grid size		27	64	125	216	343	512	729	1000
$20 \times 20 \times 20$	M_{d-64}	22	32	34	41	45	55	60	67
	M_{sp-64}	23	34	39	47	49	62	70	76
	$M_{\phi LeaS}$	27	37	42	51	54	68	75	85

Memory saving (Gb)

subdomain size	8 kdof	16 kdof	27 kdof	43 kdof	64 kdof	91 kdof
Explicit Schur	0.13	0.34	0.74	1.40	2.34	4.18
pILUt	0.02	0.04	0.09	0.14	0.29	0.47

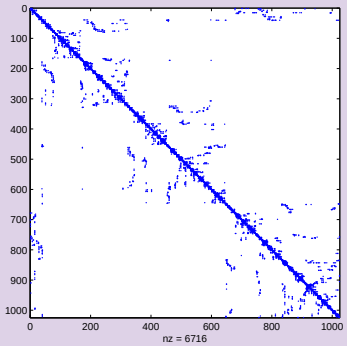
PHyLeaS next steps

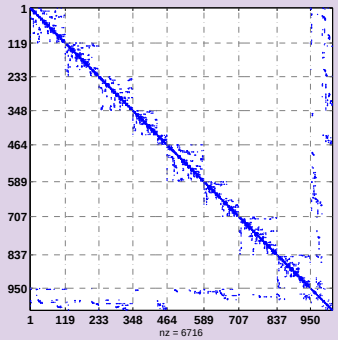
Next steps

- More experiments
- Study the effect of the interior ordering on numerical performance (quality of Schur approximation)
- Study variants of ILU to approximate the Schur complement (ILU-pack)joint work with M. Bollhoefer

Black-box hybrid method based on algebraic approach

Partitioning general matrix into N blocks





Preliminary results

Matrix			Preconditioner	# blocks				
name	size	nnz		8	16	32	64	96
bcsstk18	11,948	149,090	Block Jacobi	88	135	171	192	208
			Additive Schwarz	26	42	60	83	86
nasasrb	54,870	1,366,097	Block Jacobi	72	189	649	885	-
			Additive Schwarz	42	97	148	165	251
ex11	16,614	1,096,948	Block Jacobi	266	656	931	-	-
			Additive Schwarz	17	17	35	43	57

Consumming energy during the thesis

CPU hours consuming and CO_2 emission

- During this thesis more than 11000 runs on HPC machine:

Consumption		Pollution	
173068 h on Blue Gene @ CERFACS → 1436 Kw 10034 h on IBM JS21 @ CERFACS → 1034 Kw 3887 h on CRAY XD1 @ CERFACS → 400 Kw	2870 Kw	100 Kg of CO_2 using nuclear energy 1435 Kg of CO_2 using fuel energy 3444 Kg of CO_2 using carbon energy	Europe
85032 h on SYSTEM X @ Virg Tech → 8928 Kw	8928 Kw	312 Kg of CO_2 using nuclear energy 4464 Kg of CO_2 using fuel energy 10713Kg of CO_2 using carbon energy	USA

Thank you for your attention