Some investigations of an hybrid solver on unsymmetric and indefinite problems

Azzam Haidar

CERFACS, Toulouse

joint work with

Luc Giraud (N7-IRIT, France)

Gene around the world

January 29th, 2008, CERFACS, Toulouse, France

Outline

1. Motivations
2. Algebraic Additive Schwarz preconditioner
   - Introduction
   - Description of the preconditioner
   - Variant of Additive Shwarz preconditioner M_{AS}
3. Parallel numerical experiments
   - Numerical scalability

Outline

Motivations

Solution of very Large/Huge *ill*-conditioned linear systems

- 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 solvers is needed

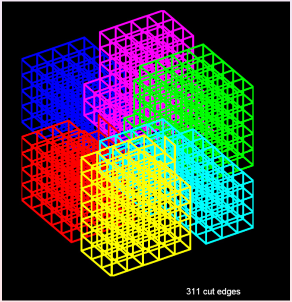
Goal

Develop a robust scalable parallel hybrid direct/iterative linear solvers

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

Non-overlapping domain decomposition

- 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 on interface



Outline

Background

Algebraic splitting and block Gaussian elimination: N sub-domains case

$$\begin{pmatrix} A_{I_1 I_1} & \dots & 0 & A_{I_1 \Gamma_1} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & A_{I_N I_N} & A_{I_N \Gamma_N} \\ A_{\Gamma_1 I_1} & \dots & A_{\Gamma_N I_N} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_{I_1} \\ \vdots \\ u_{I_N} \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_{I_1} \\ \vdots \\ f_{I_N} \\ f_\Gamma \end{pmatrix}$$

$$S u_\Gamma = \left(\sum_{i=1}^{N} R_{\Gamma_i}^T S^{(i)} R_{\Gamma_i} \right) u_\Gamma = f_\Gamma - \sum_{i=1}^{N} R_{\Gamma_i}^T A_{\Gamma_i I_i} A_{I_i I_i}^{-1} f_{I_i}$$

where $\quad S^{(i)} = A_{\Gamma_i \Gamma_i}^{(i)} - A_{\Gamma_i I_i} A_{I_i I_i}^{-1} A_{I_i \Gamma_i}$

Additive Schwarz preconditioner [Carvalho, Giraud, Meurant, 01]

Preconditionner properties

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

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

Assembled local Schur complement

local Schur complement

$$S_{mm} = \sum_{j \in AJA(m)} S_{mm}^{(j)}$$

Parallel implementation for solving $Au = f$

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

$$A^{(i)} \equiv \begin{pmatrix} A_{I_iI_i} & A_{I_i\Gamma_i} \\ A_{I_i\Gamma_i} & A_{\Gamma_i\Gamma_i} \end{pmatrix}$$

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

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

- The reduced system $Sx = b$ is solved using a distributed Krylov solver
  - One matrix vector product per iteration each processor compute $S^{(i)}(x^{(i)})^k = (y^{(i)})^k$
  - One local preconditioner apply $(\bar{S}^{(i)})^{-1}(z^{(i)})^k = (r^{(i)})^k$
  - Local neighbor-neighbor communication per iteration
  - Dot products per iteration (reduction)
- Compute simultaneously the solution for the interior unknowns $A_{I_iI_i}u_{I_i} = f_{I_i} - A_{I_i\Gamma_i}u_{\Gamma_i}$

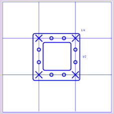
What tricks exist to construct cheaper preconditioners

Sparsification strategy

- Allow entries whose magnitude exceeds a ”drop tolerance”

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

Two-level preconditioner



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

Outline

Computational framework

Target computer

- IBM-SP4 @ CERFACS (216 procs)
- Blue Gene @ CERFACS (2048 procs)
- System X @ VIRGINIA TECH (2200 procs)

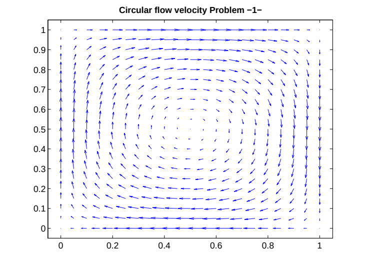
Local direct solver : MUMPS [Amestoy, Duff, Koster, L'Excellent - 01]

- 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
  `www.enseeiht.fr/apo/MUMPS - mumps@cerfacs.fr`

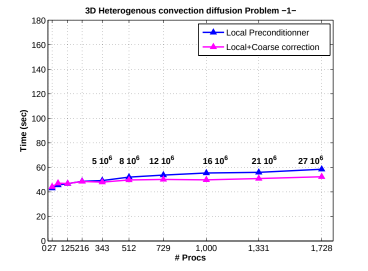
Numerical scalability for *3D* unsymmetric problems

$$-div(K.\nabla\phi) + v.\nabla\phi = f$$

xy plan view of the circular velocity field



Heterogenous convection diffusion

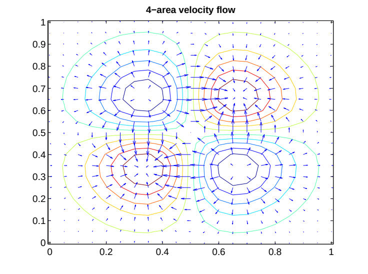


- Scaled experiments on constant subdomain size of about 15000 dof
- The computing time increases slightly when increasing # sub-domains

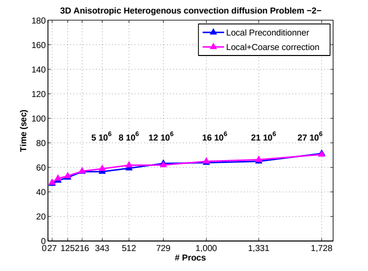
Numerical scalability for *3D* unsymmetric problems

$$-div(K.\nabla\phi) + v.\nabla\phi = f$$

xy plan view of the 4-area velocity field



Anisotropic Heterogenous convection diffusion

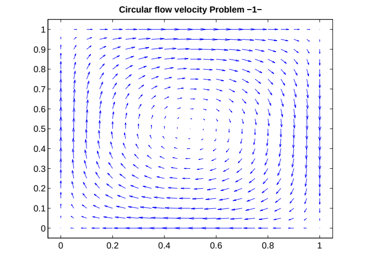


- Scaled experiments on constant subdomain size of about 15000 dof
- The computing time increases slightly when increasing # sub-domains

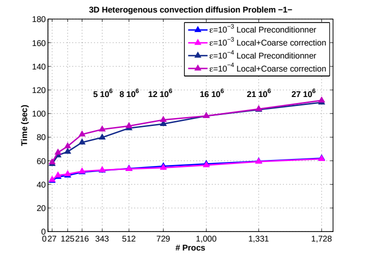
Numerical scalability for *3D* unsymmetric problems

$$-\epsilon div(K.\nabla\phi) + v.\nabla\phi = f$$

xy plan view of the circular velocity field



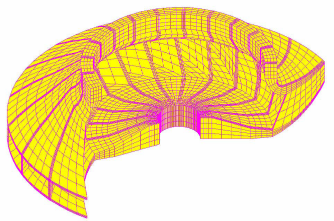
Dominated convection on heterogenous convection diffusion

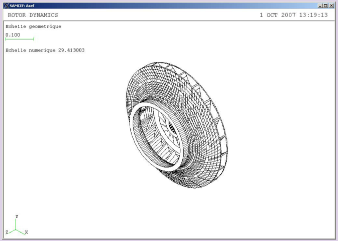


- Scaled experiments on constant subdomain size of about 15000 dof
- Smaller ϵ, harder the solution is

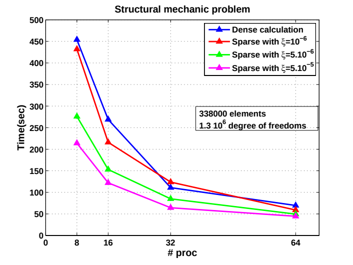
Parallel performance for *3D* indefinite problems joint work with S. Pralet, SAMTECH

Structural mechanic indefinite pb





Parallel Performance and Scalability

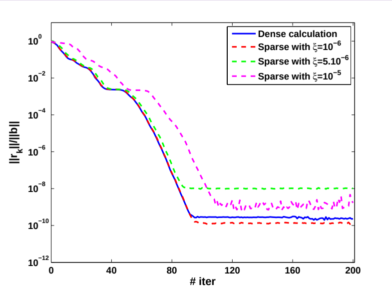


# processors	8	16	32	64
# iter Dense	59	85	106	156
# iter Sparse $\xi = 10^{-6}$	59	85	108	157
# iter Sparse $\xi = 5.10^{-6}$	60	91	114	162
# iter Sparse $\xi = 5.10^{-5}$	70	104	131	191

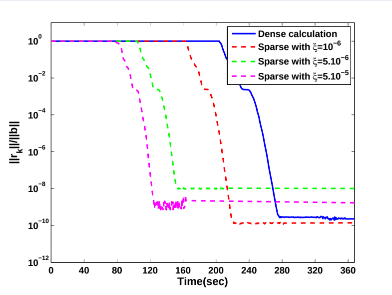
Backward error history

- Dense and Sparse preconditioners behavior

3D indefinite Rouet problem



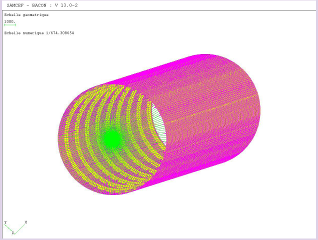
3D indefinite Rouet problem

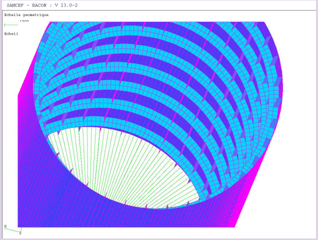


- Time = local direct factorisation + setup preconditioner + iterative loop
- Backward history on the Schur complement of a problem with 1.3 10^{6} dof mapped on 16 processors

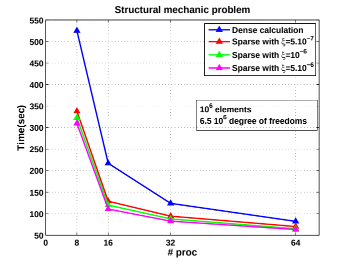
Parallel performance for *3D* indefinite problems joint work with S. Pralet, SAMTECH

Structural mechanic problem





Parallel Performance and Scalability

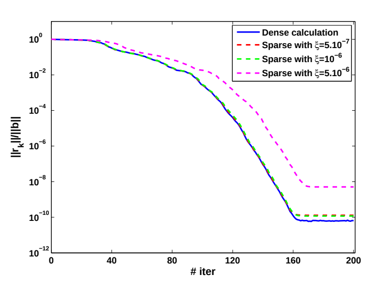


# processors	8	16	32	64
# iter Dense	98	147	176	226
# iter Sparse $\xi = 5.10^{-7}$	99	147	176	226
# iter Sparse $\xi = 10^{-6}$	101	148	177	226
# iter Sparse $\xi = 5.10^{-6}$	121	166	194	252

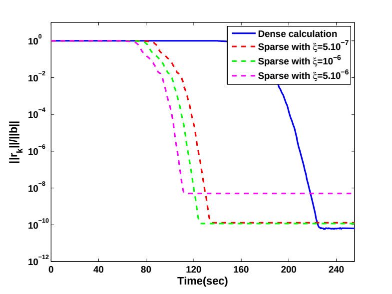
Backward error history

- Dense and Sparse preconditioners behavior

3D indefinite Fuselage problem



3D indefinite Fuselage problem



- Time = local direct factorisation + setup preconditioner + iterative loop
- Backward history on the Schur complement of a problem with 6.5 10^6 dof mapped on 16 processors