Toward robust hybrid parallel sparse solvers for large scale applications

Luc Giraud (INPT/INRIA)
joint work with Azzam Haidar (CERFACS-INPT/IRIT) and Jean Roman (ENSEIRB, LaBRI and INRIA)

1st workshop of the joint UIUC/INRIA Lab
Paris - June 11th, 2009

Outline

1. Scientific context
2. Hybrid solvers overview
3. Weak scalability on $3D$ academic model problems
4. Strong scalability on structural mechanics problems

High-end Parallel Algorithms for Challenging numerical Simulations

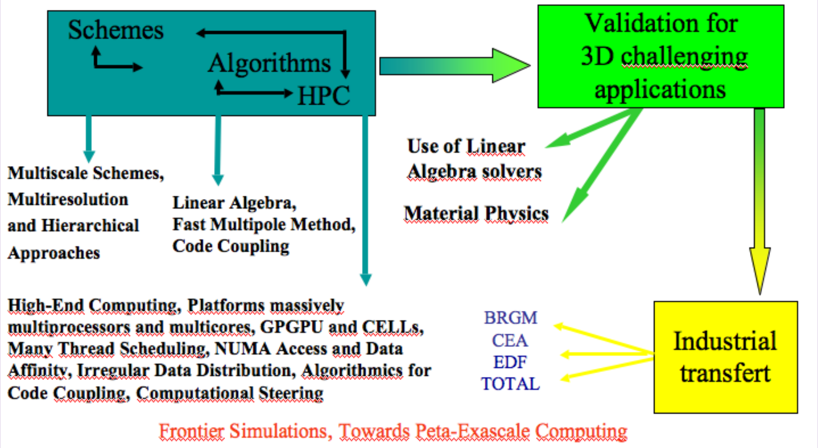
Projet Leader: Jean Roman

INRIA Bordeaux - Sud-Ouest
PRES de Bordeaux
CNRS (LaBRI UMR 5800)

Research Action INRIA-CERFACS Joint Centre

HiePACS overview

A multidisciplinary approach



General High-Performance framework

Modern (future) platforms

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

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

New algorithmic problems

- How to achieve a high scalability with applications initially designed to run over “small” number of processors?
- How can an complex applications/algorithms handle the complex memory hierarchy and the heterogeneity?
- How deal the with the huge amount of data that will be managed by our target applications?

HiePACS overview

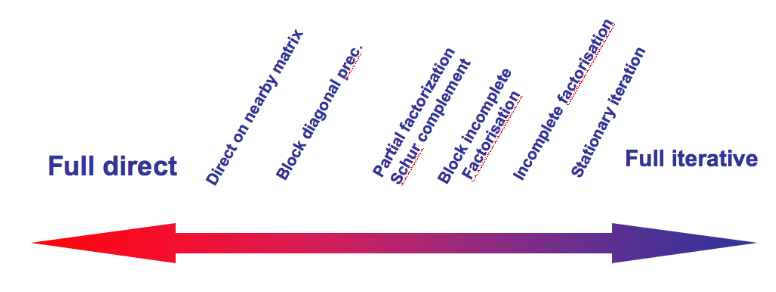
Scientific foundations

- High performance computing on next generation architectures
- High performance solvers for linear algebra problems
  - **Hybrid direct/iterative solvers based on algebraic decomposition domain**
  - Hybrid solvers based on a combination of multigrid methods and of direct solvers
  - Linear Krylov solvers
  - Eigensolvers
- High performance Fast Multipole Method for N-body problems
- Algorithmics for code coupling in complex simulations

Application domains

- Material Physics
- Application customers of high performance linear algebra solvers

Solution techniques for large linear systems



The "spectrum" of linear algebra solvers

Direct:

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

Iterative:

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

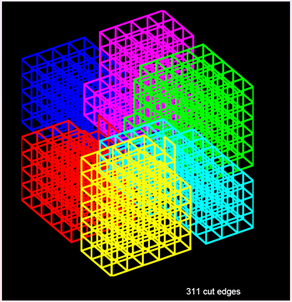
Goal

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)

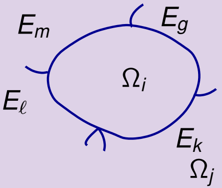
- Natural approach for PDE's
- Extend to general sparse matrices
- Partition the problem into subdomains, subgraphs
- Use a direct solver on the subdomains
- Robust preconditioned iterative solver



MaPHyS: Massively Parallel Hybrid Solver

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

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



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

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

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

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 $3D$ [PhD of A. Haidar - 08]

- 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 preserves symmetry

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]

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

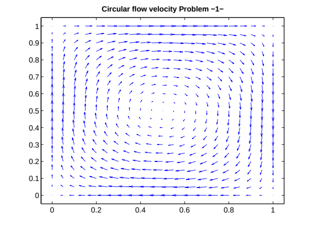
- 2048 processors
- PowerPC 440s
- 2-Way SMP
- running at 700 MHz
- 1 Gbytes/node
- latency of 1.3 - 10 μs

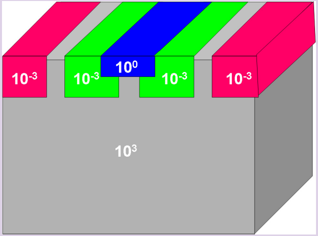
IBM JS21 @ CERFACS

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

Academic model problems

Problem patterns





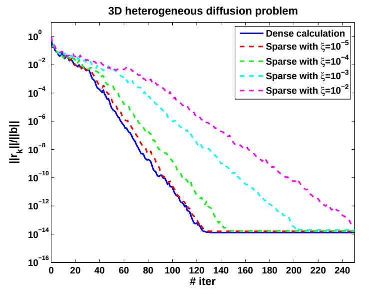
Diffusion equation ($\epsilon = 1$ and $v = 0$) and convection-diffusion equation

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

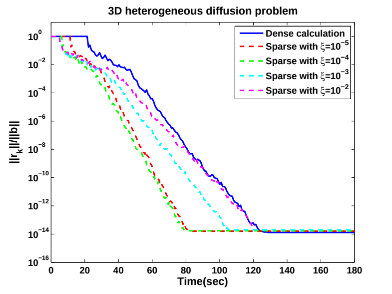
- Classical Poisson problems
- Heterogeneous problems
- Anisotropic-heterogeneous problems
- Convection dominated term

Numerical behaviour of sparse preconditioners

Convergence history of PCG



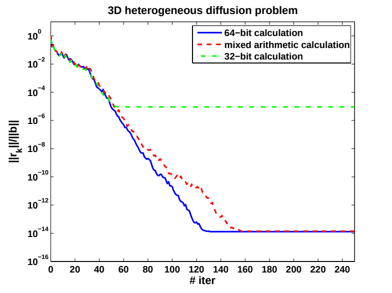
Time history of PCG



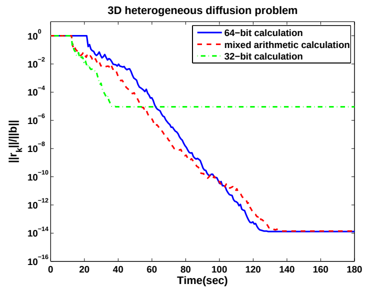
- $3D$ heterogeneous diffusion problem with 43 Mdof mapped on 1000 processors
- For ($\xi \lll$)the convergence is marginally affected while the memory saving is significant 15%
- For ($\xi \ggg$) a lot of resources are saved but the convergence becomes very poor 1%
- Even though they require more iterations,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

Numerical behaviour of mixed preconditioners

Convergence history of PCG



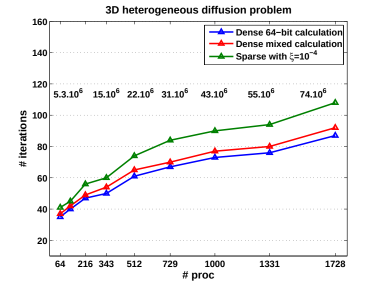
Time history of PCG



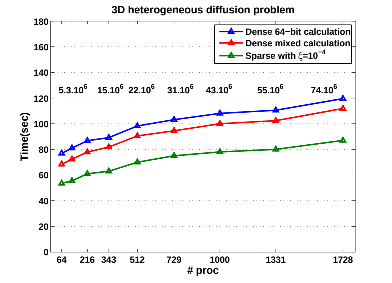
- 3D heterogeneous diffusion problem with 43 Mdof mapped on 1000 processors
- 64-bit and mixed computation both attained an accuracy at the level of 64-bit machine precision
- The number of iterations slightly increases
- The mixed approach is the fastest, down to an accuracy that is problem dependent

Weak scalability on massively parallel platforms

Numerical scalability



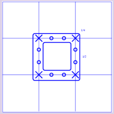
Parallel performance



- 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

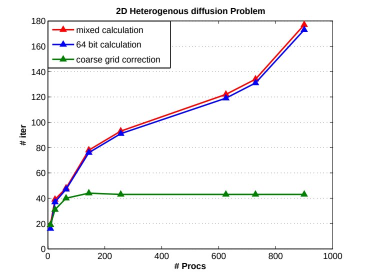
Numerical alternative: numerical scalability in $3D$

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

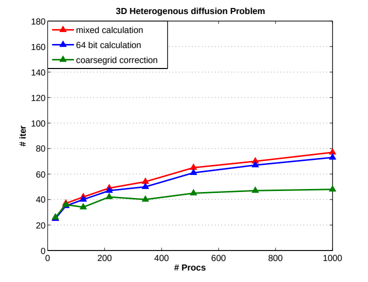


- "As many" dof in the coarse space as sub-domains [Carvalho, Giraud, Le Tallec, 01]
- Partition of unity : R_0^T simplest constant interpolation

$2D$ Heterogenous diffusion



$3D$ Heterogenous diffusion



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

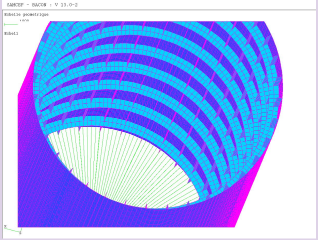
- Mixed arithmetic and 64-bit both attained an accuracy at the level of 64-bit machine precision
- Mixed preconditioner does not delay that much the convergence

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

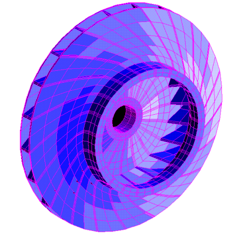
Indefinite systems in structural mechanics S.Pralet, SAMTECH

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

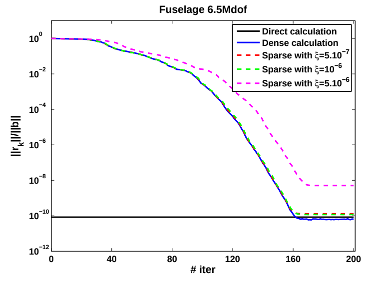
- The local matrix associated with the internal unknowns might be structurally singular
- Fix Lagrange multipliers difficulties
- Idea: enforce the Lagrange multipliers to be moved into the interface

Performance difficulties

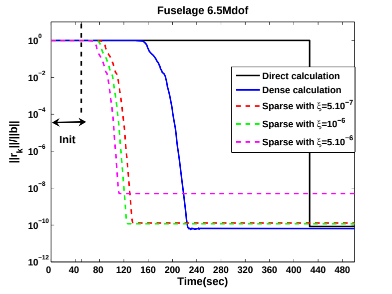
- Needs to balance and optimize the distribution of the Lagrange multipliers among the balanced subdomains
- Apply constraint (weights) to the partitioner (dual mesh graph)

Numerical behaviour of sparse preconditioners

Convergence history



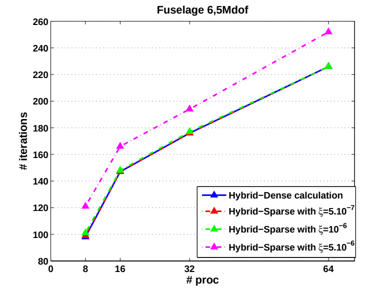
Time history



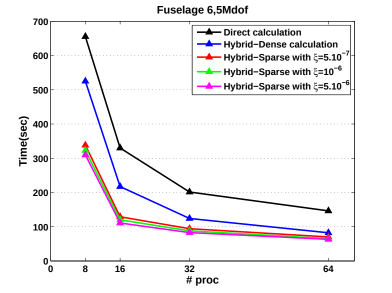
- Fuselage problem of 6.5 Mdof dof mapped on 16 processors
- The sparse preconditioner setup is 4 times faster than the dense one (19.5 v.s. 89 seconds)
- In term of global computing time, the sparse algorithm is about twice faster
- The attainable accuracy of the hybrid solver is comparable to the one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

Strong scalability

Numerical scalability



Parallel performance



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

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 3D 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
- Might be of great interest when embeded into nonlinear solver

MaPHyS: Massively Parallel Hybrid Solver

A few comments

- Robust algebraic domain decomposition with efficient parallel behaviour
- Trade-off parallel-numerics handled via multi-level parallelism (suited for clusters of SMPs)
- Cheaper memory alternatives through Schur complement approximation (on going work with Y. Saad)
- 24 month engineer support from INRIA to consolidate the prototype (ADT Parallel Scalable Hybrid Library for Large Scale Simulations: 2009-2011)

Software MaPHyS soon available through
`http://solstice.gforge.inria.fr/`

Framework: INRIA associated team / 2008-2010

PhyLeaS objectives: study of Parallel HYbrid sparse LinEAr Solvers

- Advent of massively parallel platforms requires new algorithmic designs
- Linear solvers are very often used in many large engineering simulations
- Direct and iterative approaches have assets and weaknesses: try to combine them to benefit from both

PhyLeaS partners

Univ. Minnesota, Y. Saad - TU Braunschweig, M. Bollhoefer - INRIA Bacchus, P. Hénon, P. Ramet - INRIA HiePacs, O. Coulaud, L. Giraud, J. Roman (PI) - INRIA Nachos, S. Lanteri -

Merci for your attention

Questions ?