Solveurs linéaires parallèles par décomposition de domaine algébrique

Luc Giraud

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

Journée GNR MOMAS / GDR Calcul

Paris, 5 Mai 2010

Outline

1. Background
2. A parallel algebraic domain decompostion solver
3. Parallel and numerical scalability on 3D academic problems
4. Two-level parallel implementation features and performances
5. Prospectives

Motivations

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

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

An effort for combining advantages of those solvers is needed

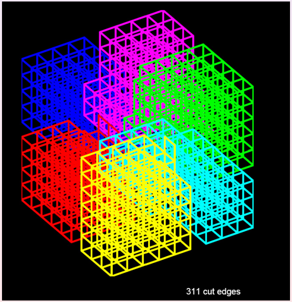
Goal: design Hybrid Linear Solvers

Develop robust scalable parallel hybrid direct/iterative linear solvers

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

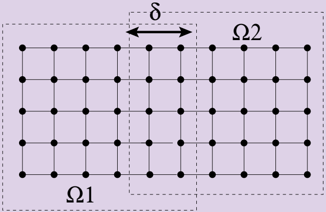
Domain Decomposition (DD)

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



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

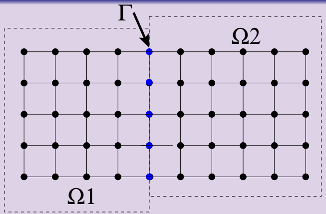
$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{1,1} & \mathcal{A}_{1,\delta} & \\ \mathcal{A}_{\delta,1} & \mathcal{A}_{\delta,\delta} & \mathcal{A}_{\delta,2} \\ & \mathcal{A}_{\delta,2} & \mathcal{A}_{2,2} \end{pmatrix} \Longrightarrow \mathcal{M}_{\mathcal{AS}}^{\delta} = \left(\boxed{\begin{matrix}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

Classical Additive Schwarz preconditioners N subdomains case

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

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 $\mathcal{A}_i x_i = b_i - \mathcal{A}_{i,\Gamma} x_\Gamma$

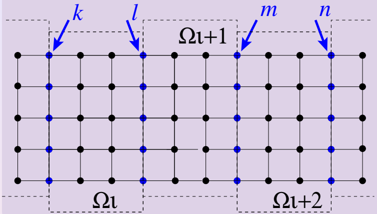
$$\begin{pmatrix} \mathcal{A}_{1,1} & 0 & \mathcal{A}_{1,\Gamma} \\ 0 & \mathcal{A}_{2,2} & \mathcal{A}_{2,\Gamma} \\ 0 & 0 & \mathcal{S} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_\Gamma - \sum_{i=1}^{2} \mathcal{A}_{\Gamma,i}\mathcal{A}_{i,i}^{-1}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

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

Schur complement reduced system



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

Distributed Schur complement

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

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

Non-overlapping Domain Decomposition

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

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

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

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

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

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

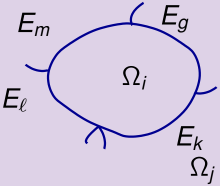
$$\searrow \quad \sum_{\iota \in adj} \mathcal{S}_{\ell\ell}^{(\iota)} \quad \nearrow$$

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

Parallel preconditioning features

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

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



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

Assembled local Schur complement

local Schur complement

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

Parallel implementation

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

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

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

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

- The reduced system $\mathcal{S}x_\Gamma = f$ is solved using a distributed Krylov solver
  - One matrix vector product per iteration each processor computes $\mathcal{S}^{(i)}(x_\Gamma^{(i)})^k = (y^{(i)})^k$
  - 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

Main characteristics in $2D$

- 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 construction of the preconditioner can be computationally expensive
- Need cheaper Algebraic Additive Schwarz form of the preconditioner

What tricks exist to construct cheaper preconditioners

Sparsification strategy through dropping

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

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

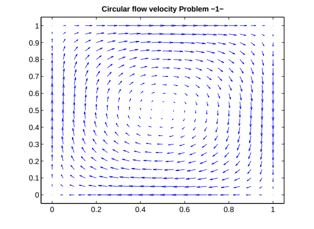
$$pILU\ (A^{(i)}) \equiv pILU \begin{pmatrix} A_{ii} & A_{i\Gamma_i} \\ A_{\Gamma_i i} & A^{(i)}_{\Gamma_i \Gamma_i} \end{pmatrix} \equiv \begin{pmatrix} \tilde{L}_i & 0 \\ A_{\Gamma i} \tilde{U}_i^{-1} & I \end{pmatrix} \begin{pmatrix} \tilde{U}_i & \tilde{L}_i^{-1} A_{i\Gamma} \\ 0 & \tilde{S}^{(i)}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

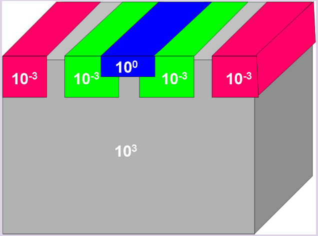
Mixed arithmetic strategy

- 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]

Academic model problems

Problem patterns





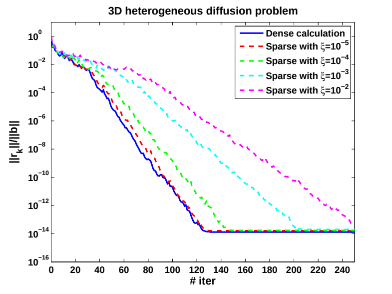
Diffusion equation ($\epsilon = 1$ and $v = 0$) and convection-diffusion equation

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

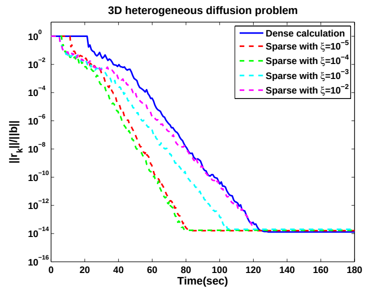
- Heterogeneous problems
- Anisotropic-heterogeneous problems
- Convection dominated term

Numerical behaviour of sparse preconditioners

Convergence history of PCG



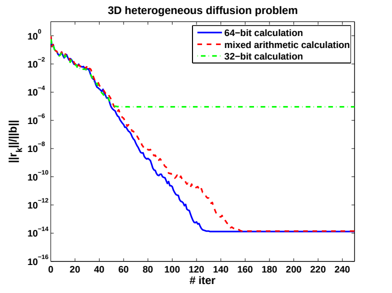
Time history of PCG



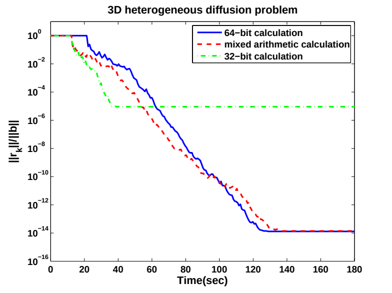
- $3D$ heterogeneous diffusion problem with 43 Mdof mapped on 1000 processors
- For ($\xi \lll$)the convergence is marginally affected while the memory saving is significant 15%
- For ($\xi \ggg$) a lot of resources are saved but the convergence becomes very poor 1%
- Even though they require more iterations, 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



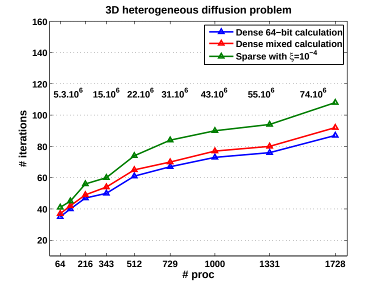
Time history of PCG



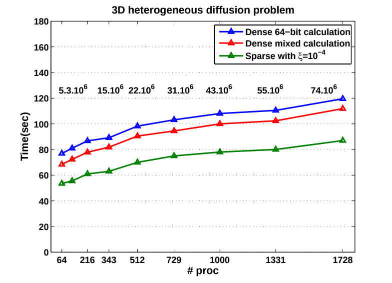
- $3D$ heterogeneous diffusion problem with 43 Mdof mapped on 1000 processors
- 64-bit and mixed computation both attained an accuracy at the level of 64-bit machine precision
- The number of iterations slightly increases
- The mixed approach is the fastest, down to an accuracy that is problem dependent

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

Summary on the model problems

[L.Giraud, A.Haidar, L.T.Watson - 08] [L.Giraud, A.Haidar, Y.Saad - 10]

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 too much the convergence

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

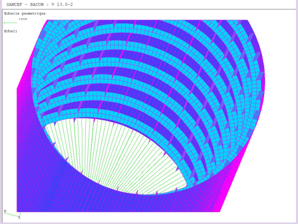
- Although these preconditioners are local, possibly not numerically scalable, they exhibit a fairly good parallel time scalability (possible fix for elliptic problems)
- The trends that have been observed on this choice of model problem have been observed on many other problems

Experiments on large $3D$ real life applications

Application areas

- Structural mechanics : real SPD and symmetric indefinite linear systems.
- Electromagnetism : complex symmetric non-Hermitian.
- Seismic : complex symmetric non-Hermitian.

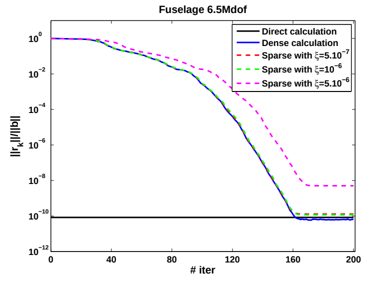
Fuselage of 6.5 Mdof



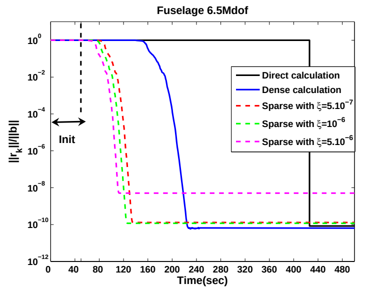
- Linear elasticity
- Composed of its skin, stringers and frames
- Midlinn shell elements are used
- Each node has 6 unknowns
- One extremity is fixed
- On the other extremity a rigid body element is added
- A force perpendicular to the axis is applied

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

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

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

Toward a “‘black-box” parallel solver

MaPHyS package- ongoing work

- Apply ideas to adjacency graph of sparse matrices no longer to meshes (ANR-CIS Solstice).
- Replace full-MPI two-level parallelism by mixed multi-threaded MPI parallel implementation to better comply with NUMA cluster features (PasTiX, Super_LU).
- Improve the solver capability for symmetric indefinite et fully unsymmetric (France-Berkeley Fund pending proposal).
- Perform a complexity analysis to study the computational scalabality

http://www.inria.fr/recherche/equipes/hiepacs.fr.html

Acknowledgments

Credit to co-workers

- Numerical methodologies:
  E. Agullo (INRIA), A. Guermouche (INRIA), A. Haidar (ICL, Univ. Tennessee), Y. Lee (INRIA), J. Roman (INRIA), Y. Saad (Univ. Minnesota). MUMPS & PaStiX developers.
- Applications:
  H. Benhadjali (SEISCOPE), D. Goudin (CEA-CESTA), S. Operto (Géosciences Azur), S. Pralet (SAMTECH), J. Virieux (LGIT).

Merci pour votre attention

Questions ?