Algebraic multilevel preconditioners on massively parallel computers

Gérard MEURANT
CEA/DIF

September 2, 2006

We would like to solve large sparse linear systems on parallel computer in a scalable way

$$Ax = b$$

with A a nonsingular sparse matrix of order n, n may be several tens or hundreds millions

We use Krylov iterative methods with algebraic multigrid preconditioners

Today, we will concentrate on symmetric systems using CG

Algebraic multilevel preconditioners

$$Ax = b$$

A symmetric positive definite $\Rightarrow$ PCG

For PCG to be scalable the preconditioner must be s.t.:

- the number of iterations is (almost) constant when the size of the problem increases

- the complexity of applying the preconditioner is proportional to n

Algebraic multigrid was introduced by J. Ruge and K. Stuben (1985)

It mimics geometric multigrid

A "grid" $\equiv$ (sub)space of unknowns (vertices) of the matrix graph

- The main difficulty is to have a method both efficient and parallel

Multilevel preconditioners (V-cycle)

Starting from the null vector :

0– on the coarsest level, solve exactly using Gauss, otherwise

1– Do ν smoothing iterations

2– Restrict the residual r to $r_c = Rr$ (next coarse level)

3– Recursively solve $A_c e_c = r_c$, $A_c = RAP$, $R = P^T$

4– Interpolate e_c to $e = Pe_c$ (next fine level)

5– Add the correction e to the current iterate

6– Do ν smoothing iterations

Smoothers

- ∘ symmetric Gauss–Seidel (not //)
- ∘ incomplete Cholesky (not //) LDL^T
  - – IC(0)
  - – IC with fill–in (values)
  - – IC with fill–in (levels)

$$LD^{-1}L^T(x^{k+1} - x^k) = b - Ax^k$$

∘ Approximate inverse AINV from M. Benzi (Emory Univ.) and al.

$$M \approx A^{-1}, \quad M = ZD^{-1}Z^T$$

where Z is upper triangular with 1 on the diagonal and D is diagonal

The parameter τ defines which elements are kept in Z as the factorization (by columns) proceeds

It works for H–matrices, for SPD matrices one uses SAINV (Stabilized AINV)

Smoother: Richardson iteration defined as

$$x^{k+1} = x^k + M(b - Ax^k)$$

Influence matrix

How to define the coarse levels?

$$\mathcal{N} = \{1, \ldots, n\}, \quad \mathcal{N} = F \cup C$$

Set of indices: standard AMG choice (Ruge-Stuben) for M–matrices

i is a row index

$$S_i = \{j | -a_{i,j} > \theta \max_{k \neq i} (-a_{i,k}), \quad \theta < 1\}$$

From S_i we construct S (matrix with 1 and 0 elements)

General case

$$S_i^A = \{j \neq i \mid |a_{i,j}| > \tau \max_k |a_{i,k}|, \quad \tau < 1\}$$

We keep at least one 1 for the largest modulus element ('b')

τ parameter to be chosen

It is usually better to symmetrically normalize the matrix

Coarsening algorithm

The “standard” algorithm is:

Weights $w_i =$ nb of points which depend on i (using S)

1- Choose a point i of maximal weight as a C point

2- Flag the points that i influences (with S) as F points

3- Add 1 to the weights of points influencing these new F points (to give them a better chance to be chosen as C points in the next steps)

4- Decrease by 1 the weights of points that depend on i

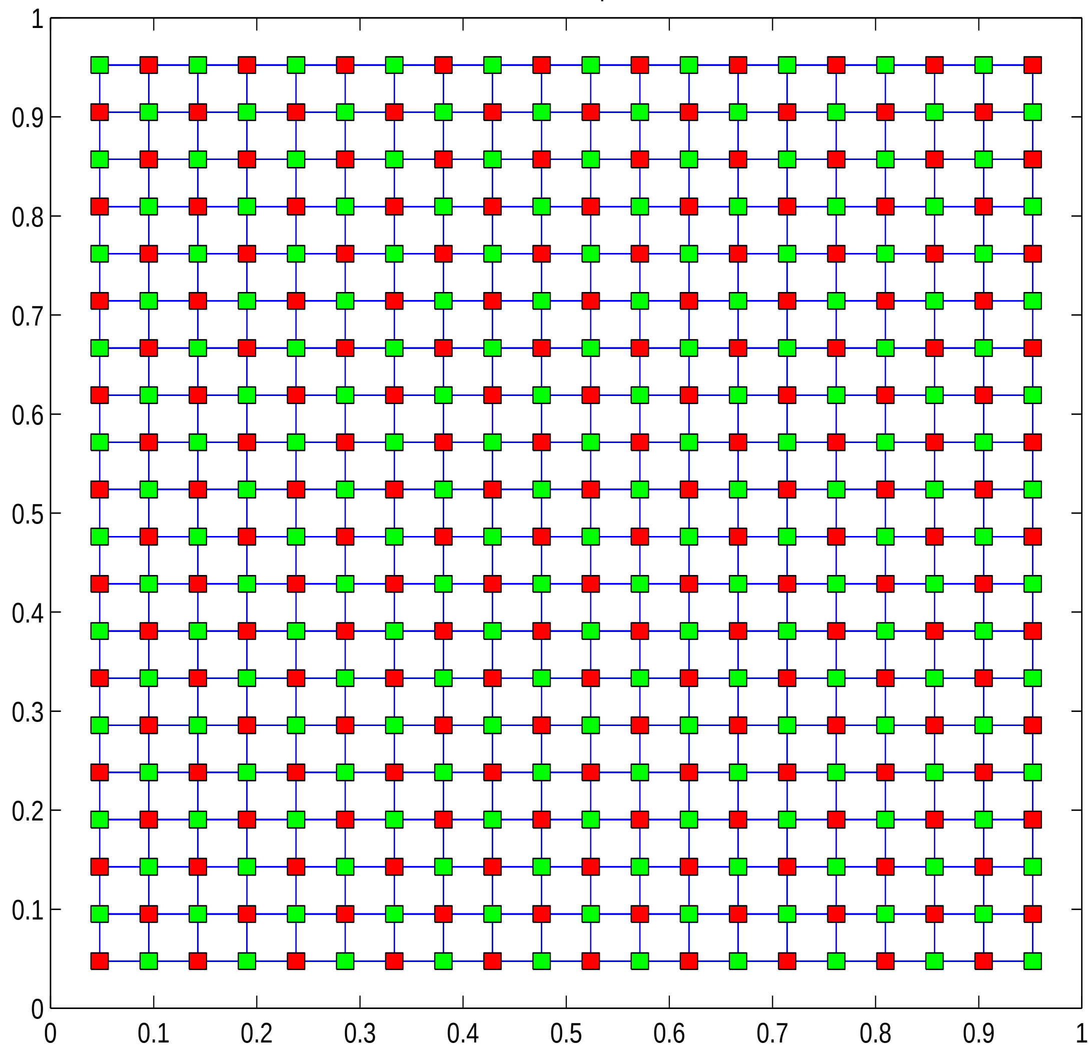
Repeat steps 1-4 until all the points are labelled

Example: Poisson equation in a square, 5-point finite differences, $n = 400$

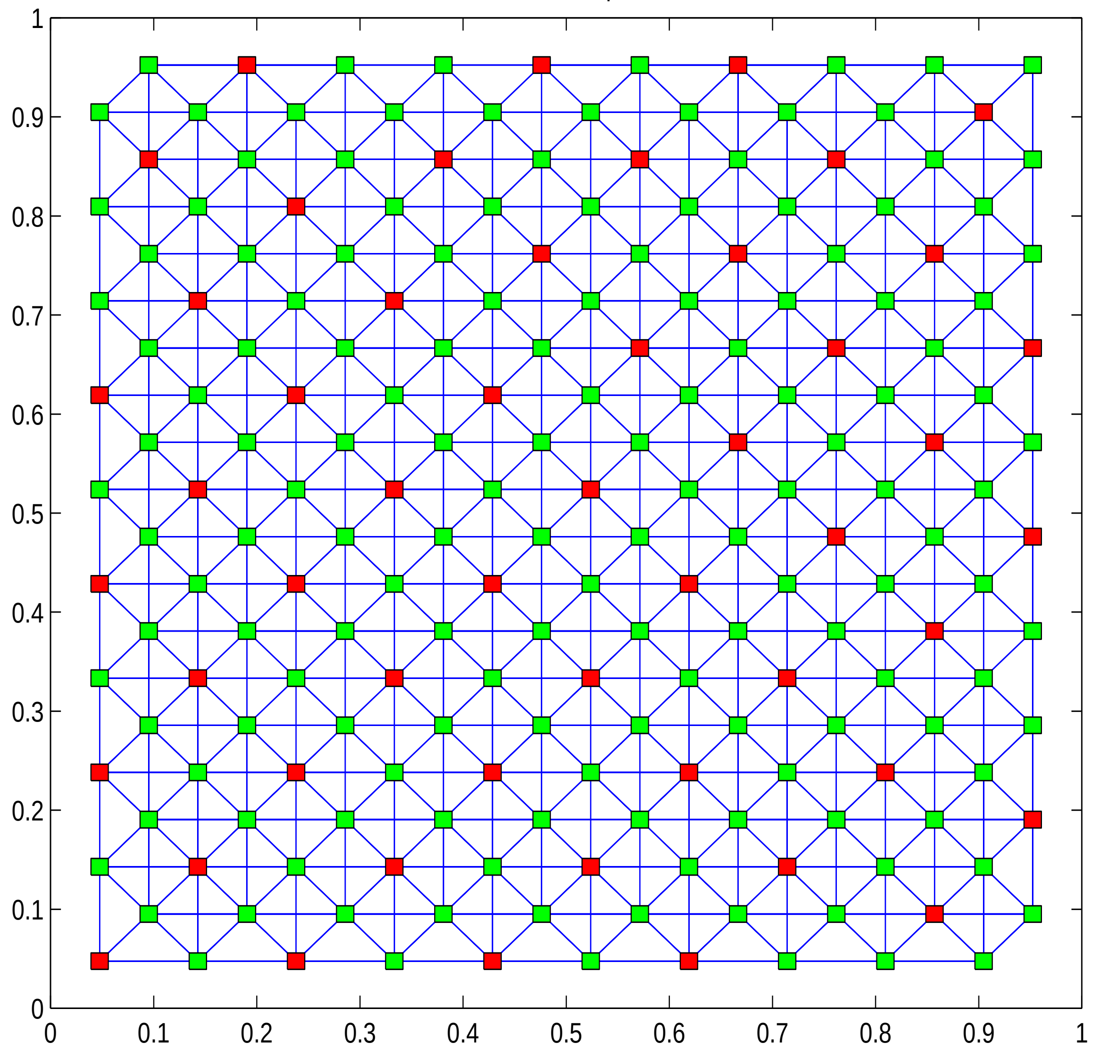
The graph of A is the same as the mesh

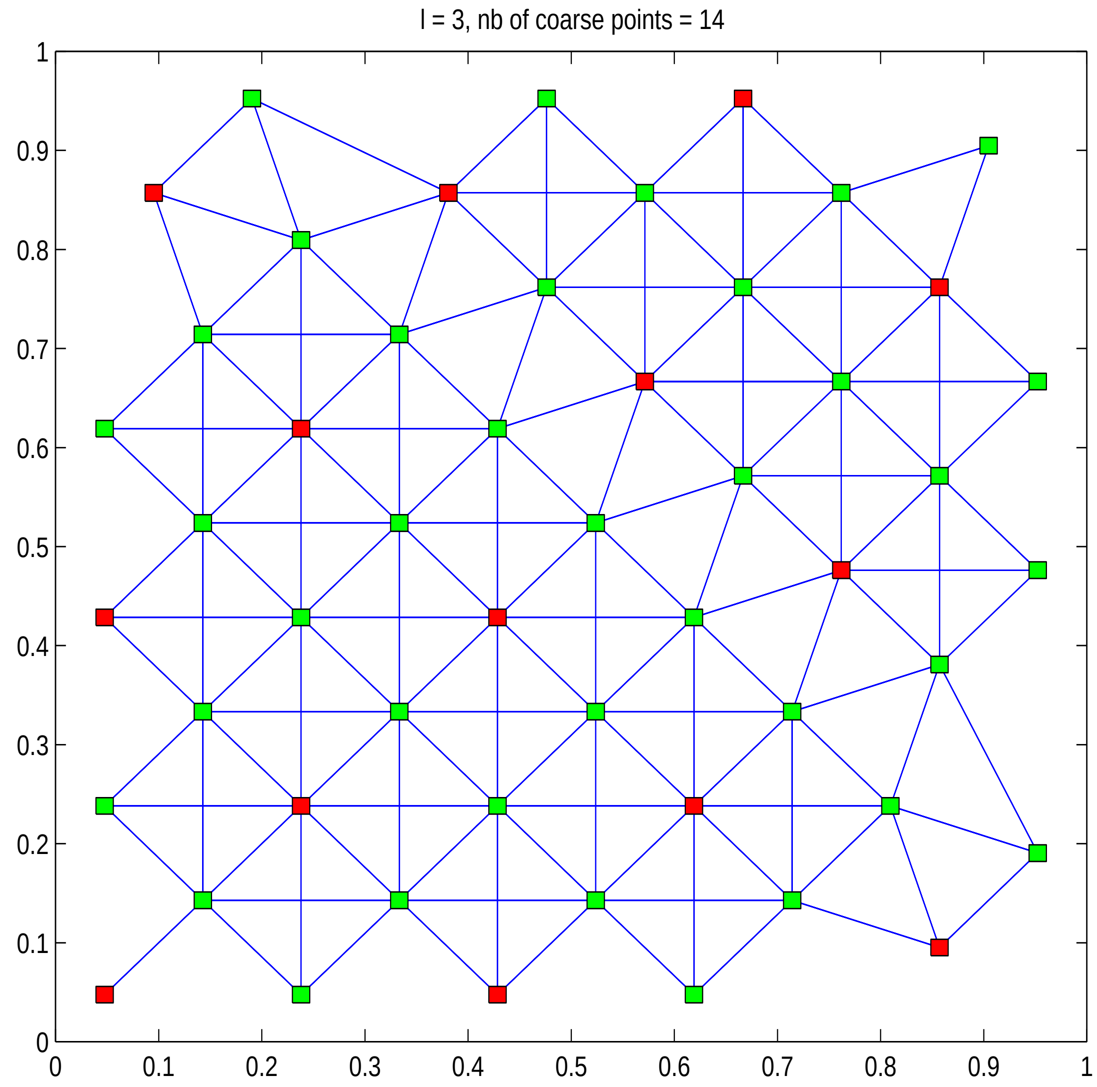
The matrix is normalized with ones on the diagonal

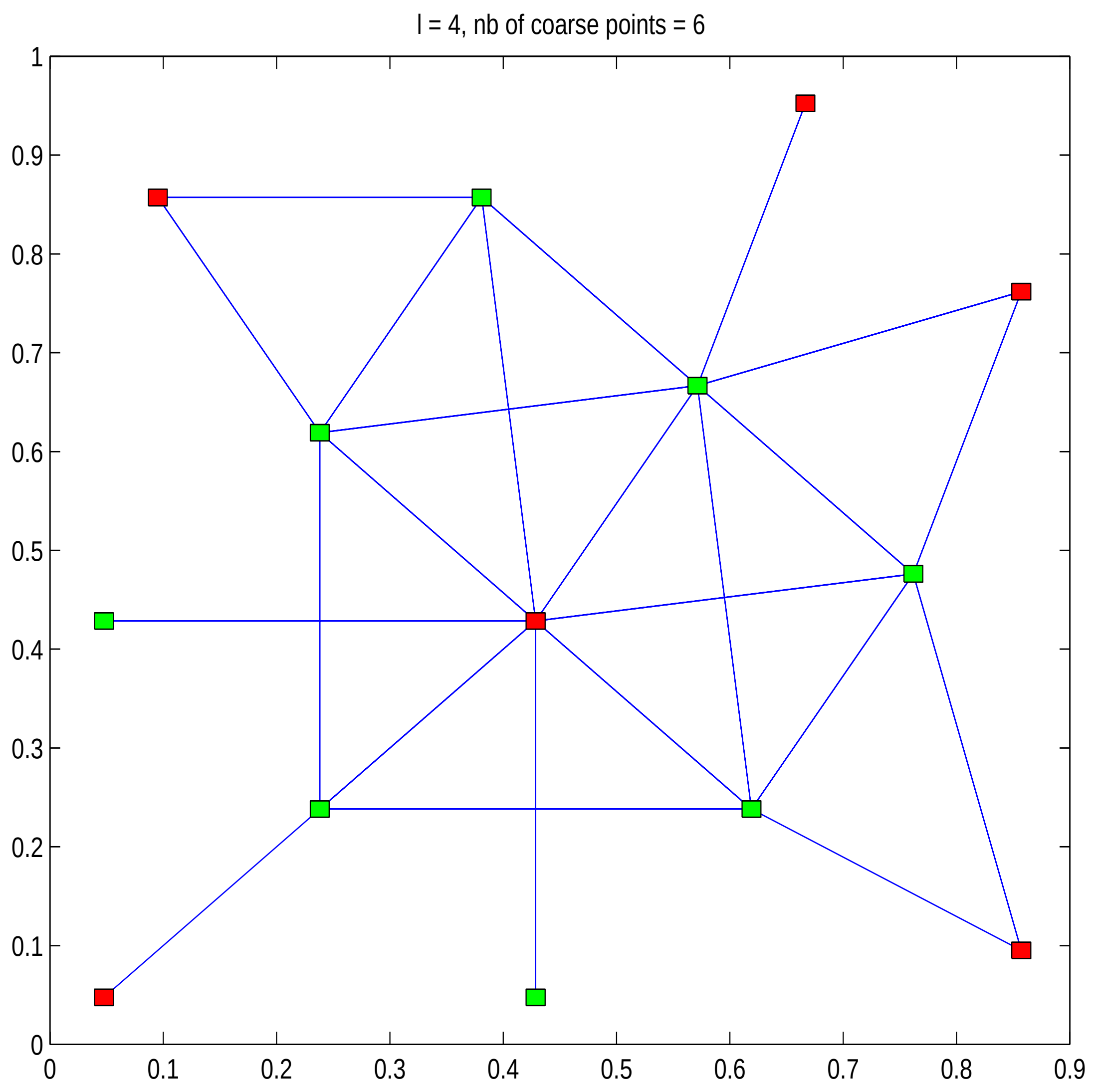
graphs and C and F points for all levels l = 1, nb of coarse points = 200



l = 2, nb of coarse points = 51




l = 3, nb of coarse points = 14
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


l = 4, nb of coarse points = 6
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Interpolation algorithm

- $i \in F, j \in C$

$$\omega_{i,j} = -\frac{a_{i,j} + \sum_{k \in D_i^S} \frac{a_{i,k} a_{k,j}}{\sum_{m \epsilon C_i} a_{k,m}}}{a_{i,i} + \sum_{k \in D_i^W} a_{i,k}}$$

D_i^S and D_i^W are strong and weak couplings

This comes from writing $Ae = 0$ and using $e_j \approx e_i$ for weak connections and a weighted average for connections with F points

Coarse matrices

The interpolation algorithm defines P and $R = P^T$

$$A_C = RAP$$

How to parallelize the smoothers?

Domain decomposition

◦ Partition the graph of A (or sometimes S) with or without overlapping (ghost nodes)

◦ symmetric Gauss–Seidel

parallelized by using Jacobi for interface nodes (SGSJ)

◦ incomplete Cholesky or AINV

parallelized by ignoring dependencies between subdomains (ICp, SAINVp)

Only the finest level is partitioned, this may cause load balancing problems on coarse levels

Parallel coarsening

◦ LLNL algorithms (Cleary, Falgout, Henson and Jones)

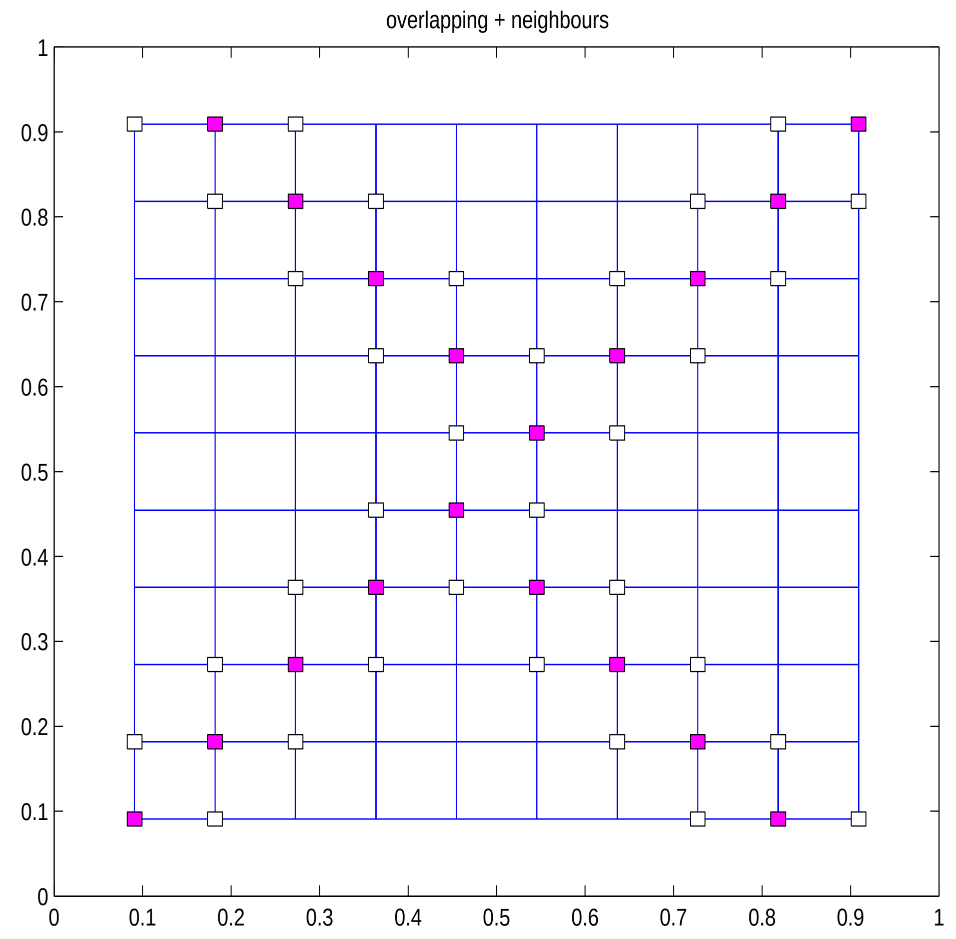
• They start on several “independent” nodes at the same time

◦ Other method :

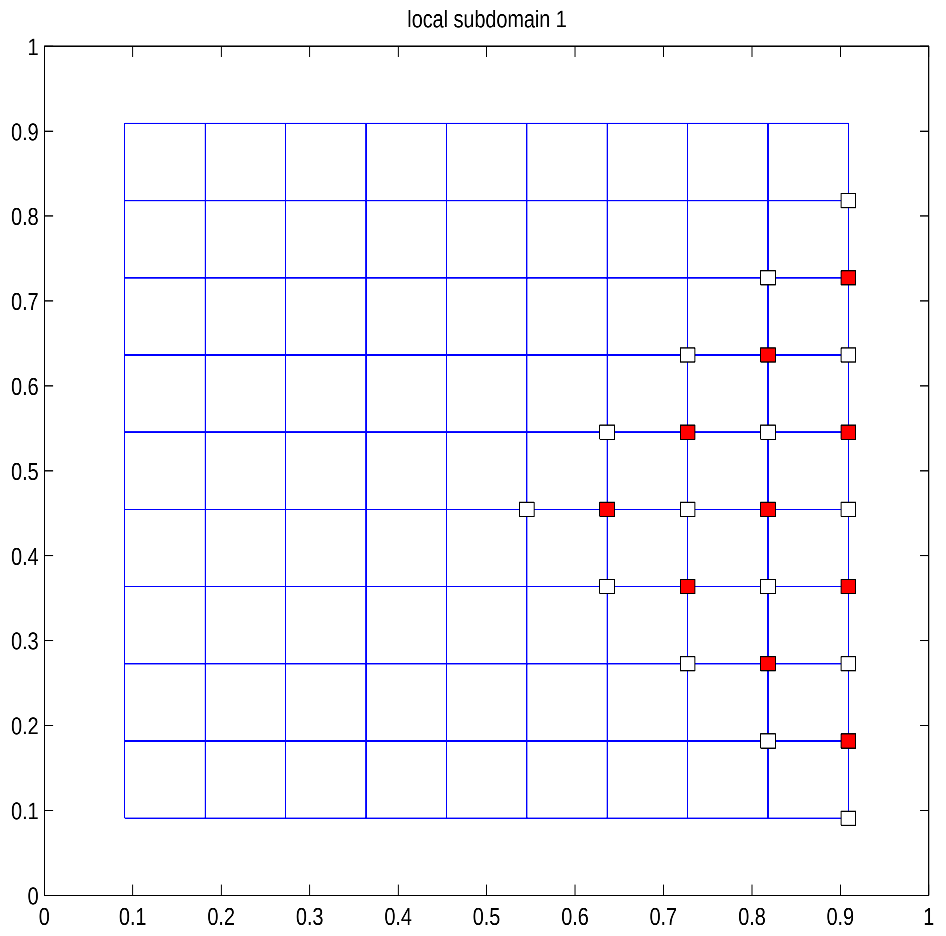
• Start by coarsening the overlapping or the interface

• Flag the (subdomain) neighbors of these C points as F points (without introducing $F - F$ connections)

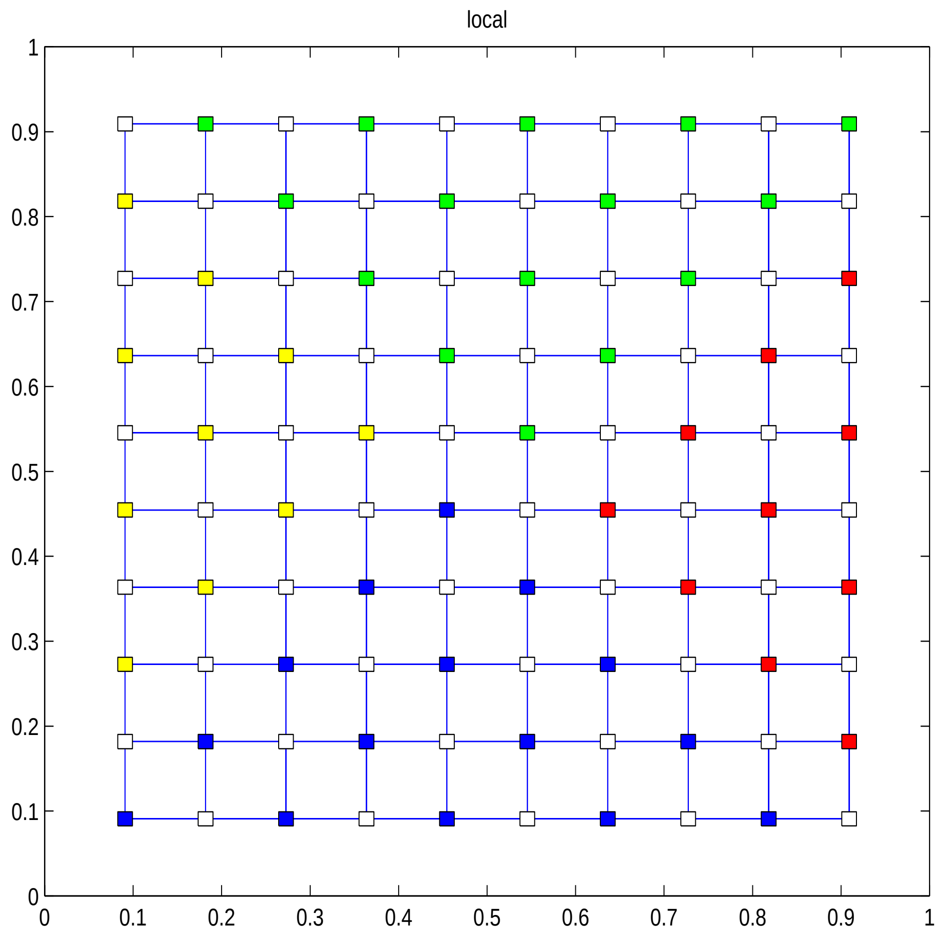
• Coarsen the subdomains using the preceding step as “boundary conditions”



Interface (magenta) + neighbors (white), 4 subdomains



First subdomain



Global result, 4 subdomains

Numerical results

5 point finite differences, unit square, $m \times m$ mesh

b random, $x^0 = 0$

stopping criterion: $\|r^k\| \leq 10^{-10}\|r^0\|$

Small sequential problems

PCG, Poisson, $\tau = 0.06$, (‘ic’, ‘a’, ‘st’, ‘st’), $\gamma = 1$

m	$\nu = 1$	$\nu = 2$

40	5
	op=1598145, /n=998.8
	str=35667, /n=22.3
	$\kappa = 1.03$
		5
		op=2631408, /n=1644
		$\kappa = 1.01$

50	5
	op=2528438, /n=1011
	str=56497, /n=22.6
	$\kappa = 1.02$
		5
		op=4165773, /n=1666
		$\kappa = 1.01$

60	6
	op=4265230, /n=1185
	str=81388, /n=22.6
	$\kappa = 1.03$
		5
		op=6008813, /n=1669
		$\kappa = 1.01$

◦ Parallel version with no fill–in between subdomains, incomplete Cholesky

PCG, Poisson, $m = 40$, $\tau = 0.05$, ('id', 'b', 'sd', 'st'), without $F - F$ connections on the fine level

nb sd	nb it	flops	storage
1	5	1 598 253	35 550
2	8	2 484 338	36 790
4	7	2 186 602	36 555
8	8	2 517 864	37 529
16	9	2 824 986	37 545
32	11	3 586 071	34 670

◦ AINV

PCG, Poisson, $m = 40$, $\tau = 0.05$, (‘ad’, ‘b’, ‘sd’, ‘st’) without $F - F$ connections on the fine level

nb sd	nb it	flops	storage
1	14	3 929 793	36 005
2	15	4 277 693	36 882
4	14	3 971 013	36 611
8	13	3 749 853	37 236
16	12	3 502 164	37 133
32	12	3 750 659	34 536

Numerical results on the CEA TERA 1

(HP-Compaq, Alpha EV6 processors 1Ghz)

5 (7) point finite differences, unit square (cube), $m \times m$ mesh, b random

$x^0 = 0$

stopping criterion $\|r^k\| \leq 10^{-10}\|r^0\|$

Domain decomposition with squares (cubes), $m_p^{2(3)}$ unknowns per processor

- A is distributed by rows

- Poisson equation
- Diffusion problem with discontinuous and anisotropic coeff

Partition of $[0,1]^2$ in 4 squares

diffusion coeff $(1,1),(10,100),(100,10),(1000,1000)$

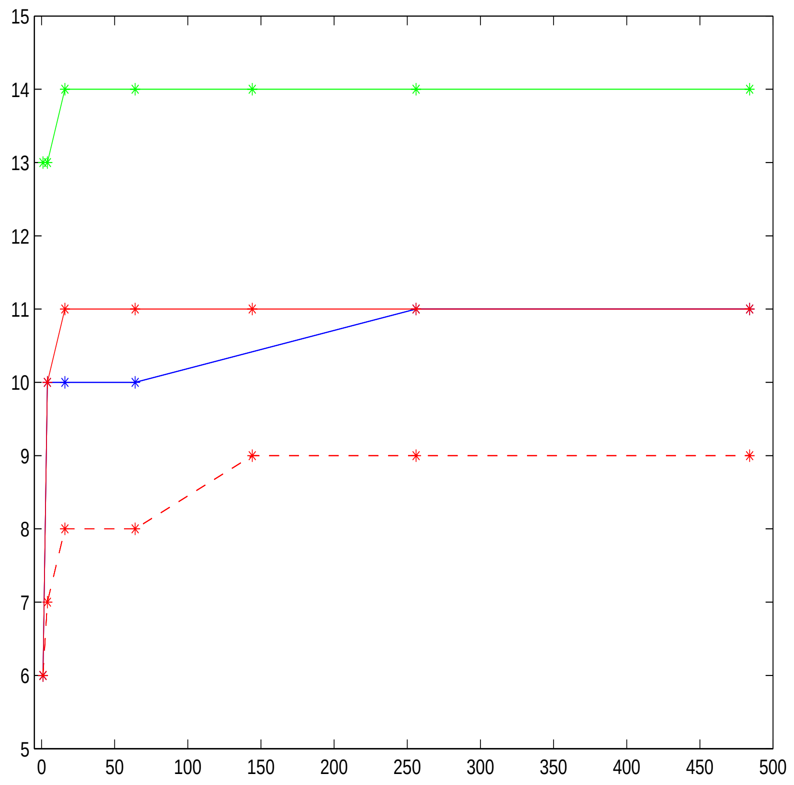
First experiment

- Poisson

$m_p = 250 \longrightarrow 62500$ unknowns per processor,

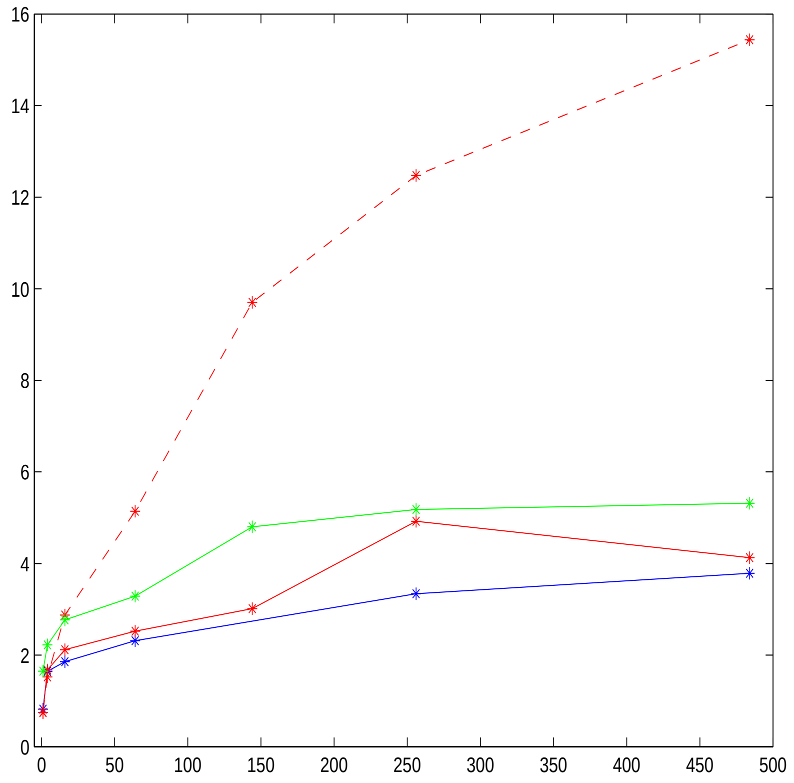
$p = 1, 4, 16, 64, 144, 256, 484$

- largest problem has $\simeq 30\ 10^6$ unknowns



Nb of iterations for Poisson as a function of p

coarsening : LLNL, blue: SGSJ, red dashed: IC, red: ICp, green: SAINVp



“elapsed” time (s) for Poisson as a function of p

coarsening : LLNL, blue: SGSJ, red dashed: IC, red: ICp, green: SAINVp

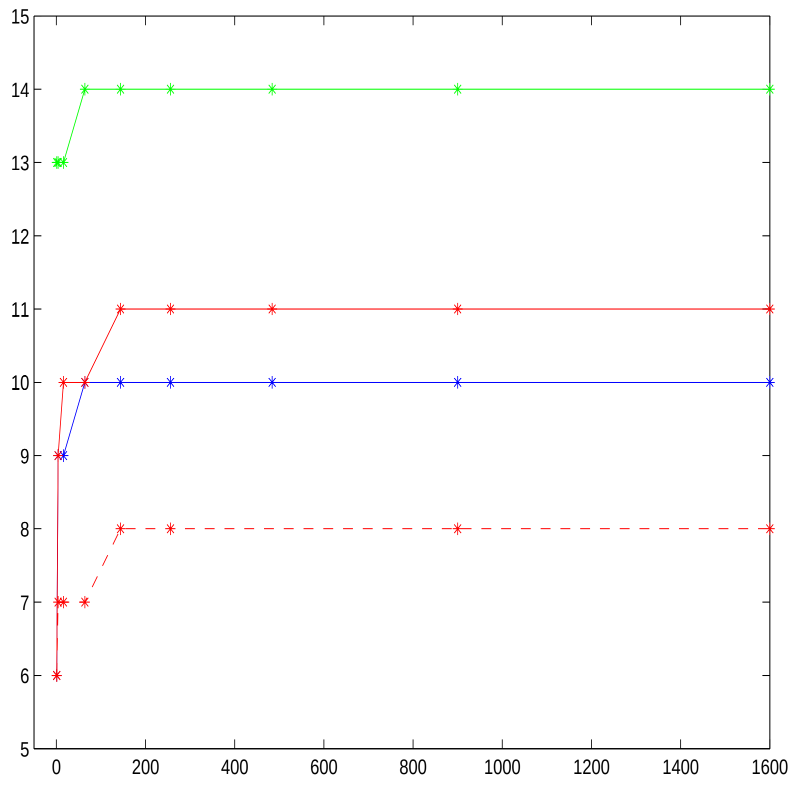
Second experiment

- Poisson

$m_p = 100 \longrightarrow 10000$ unknowns per processor

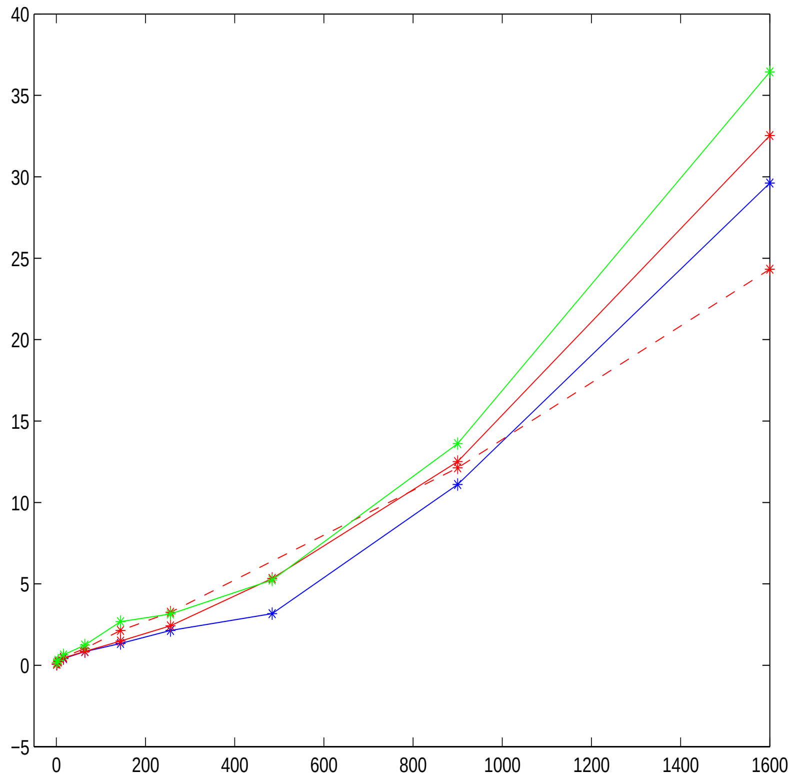
$p = 1, 4, 16, 64, 144, 256, 484, 900, 1600$

- Largest problem has $16\ 10^6$ unknowns



Nb of iterations for Poisson as a function of p

coarsening : LLNL, blue: SGSJ, red dashed: IC, red: ICp, green: SAINVp



“elapsed” time (s) for Poisson as a function of p

coarsening : LLNL, blue: SGSJ, red dashed: IC, red: ICp, green: SAINVp

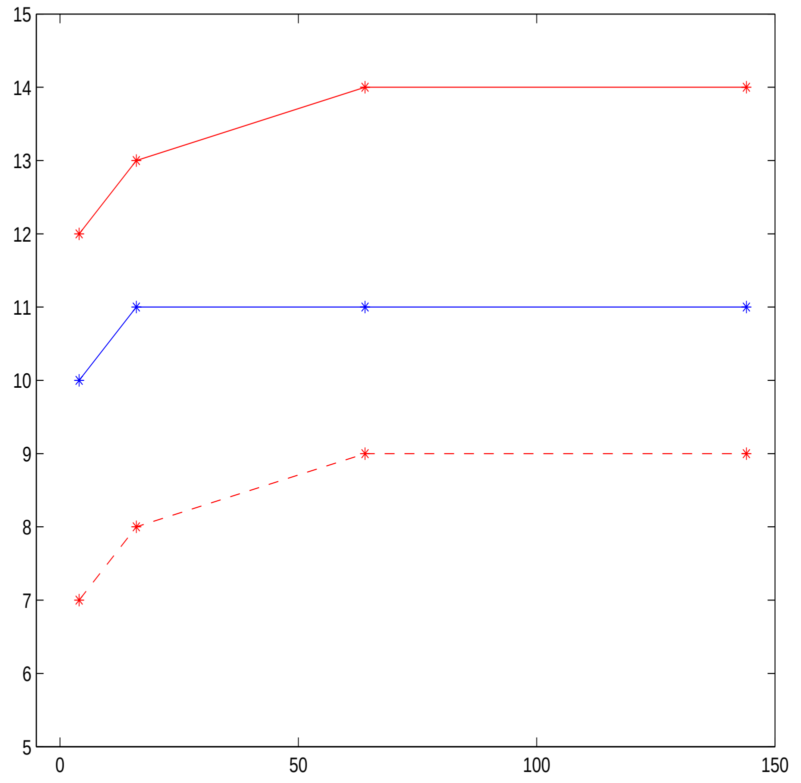
Third experiment

- Discontinuous and anisotropic problem

$m_p = 250 \longrightarrow 62500$ unknowns per processor

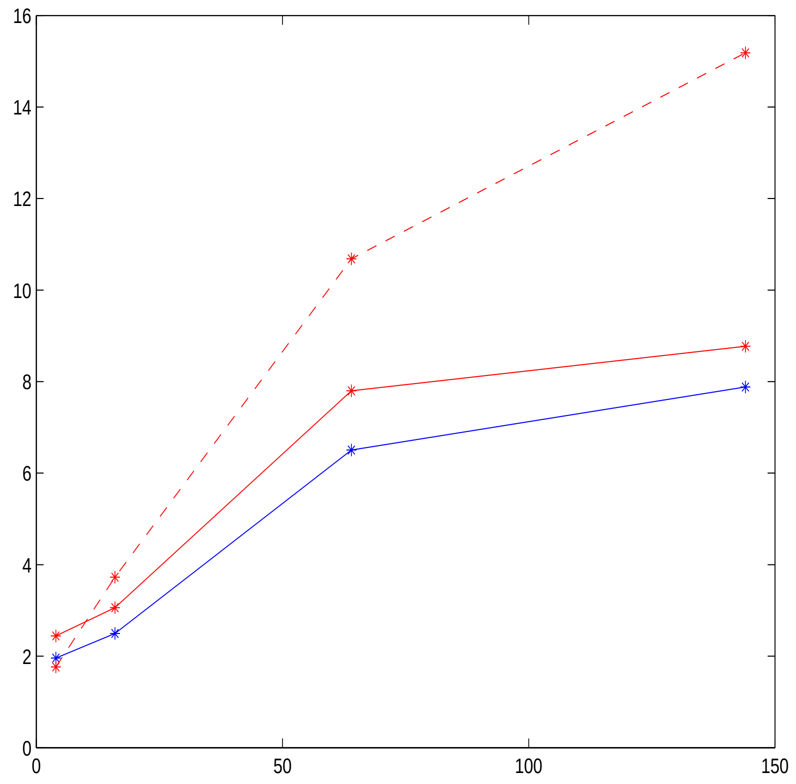
$p = 4, 16, 64, 144$

- Largest problem has $4\ 10^6$ unknowns



Nb of iterations for the discontinuous and anisotropic pb as a function of p

coarsening : LLNL, blue: SGSJ, red dashed: IC, red: ICp



“elapsed” time (s) for the discontinuous and anisotropic pb as a function of p

coarsening : LLNL, blue: SGSJ, red dashed: IC, red: ICp

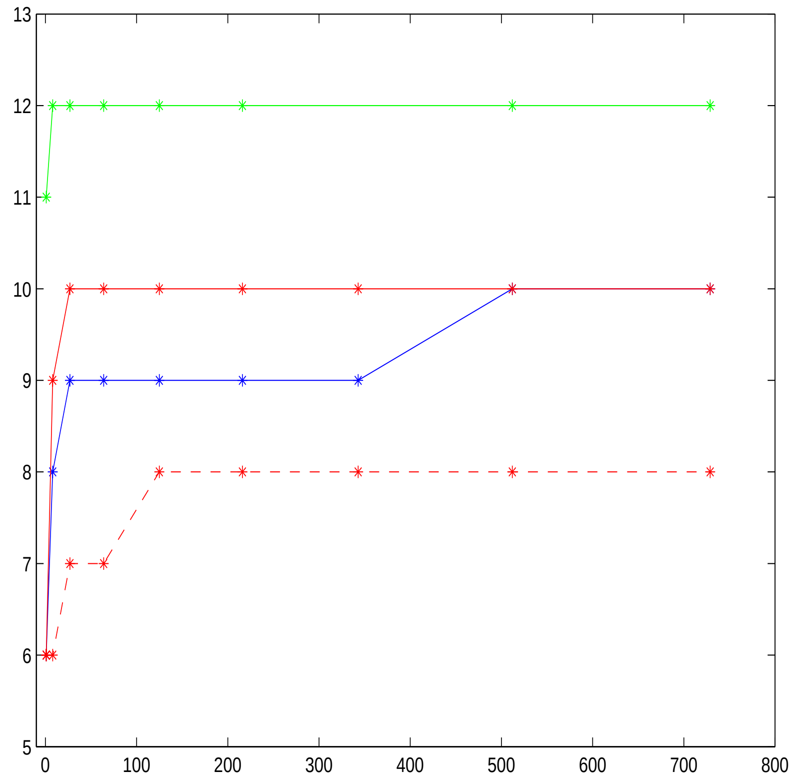
Fourth experiment

- 3D Poisson in $[0,1]^3$

$m_p = 30 \to m_p^3 = 27000$ unknowns per processor,

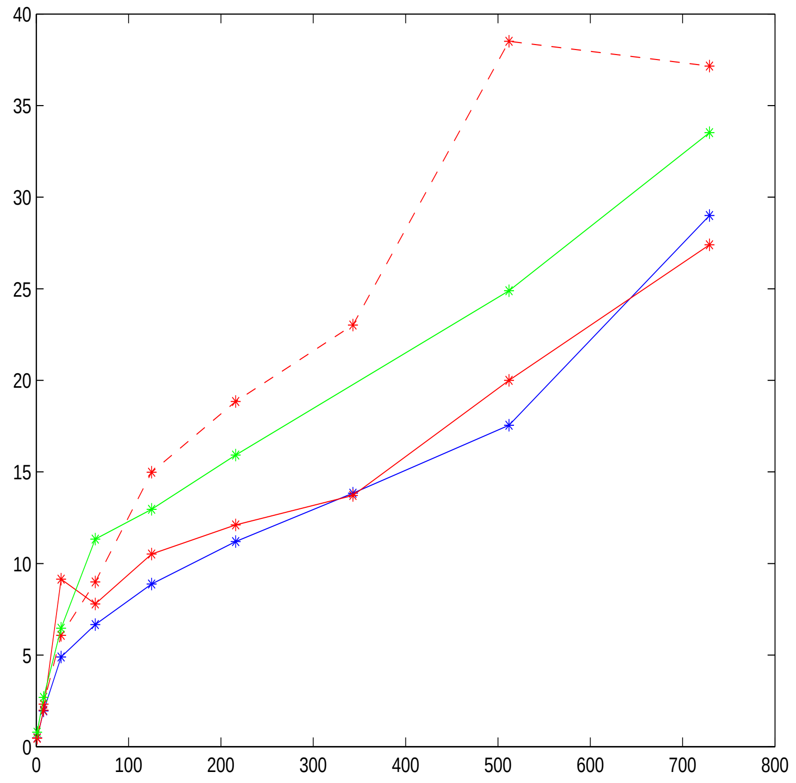
$p = 1, 8, 27, 64, 125, 216, 343, 512, 729$

- Largest problem has $\simeq 20\ 10^6$ unknowns



Nb of iterations for 3D Poisson as a function of p

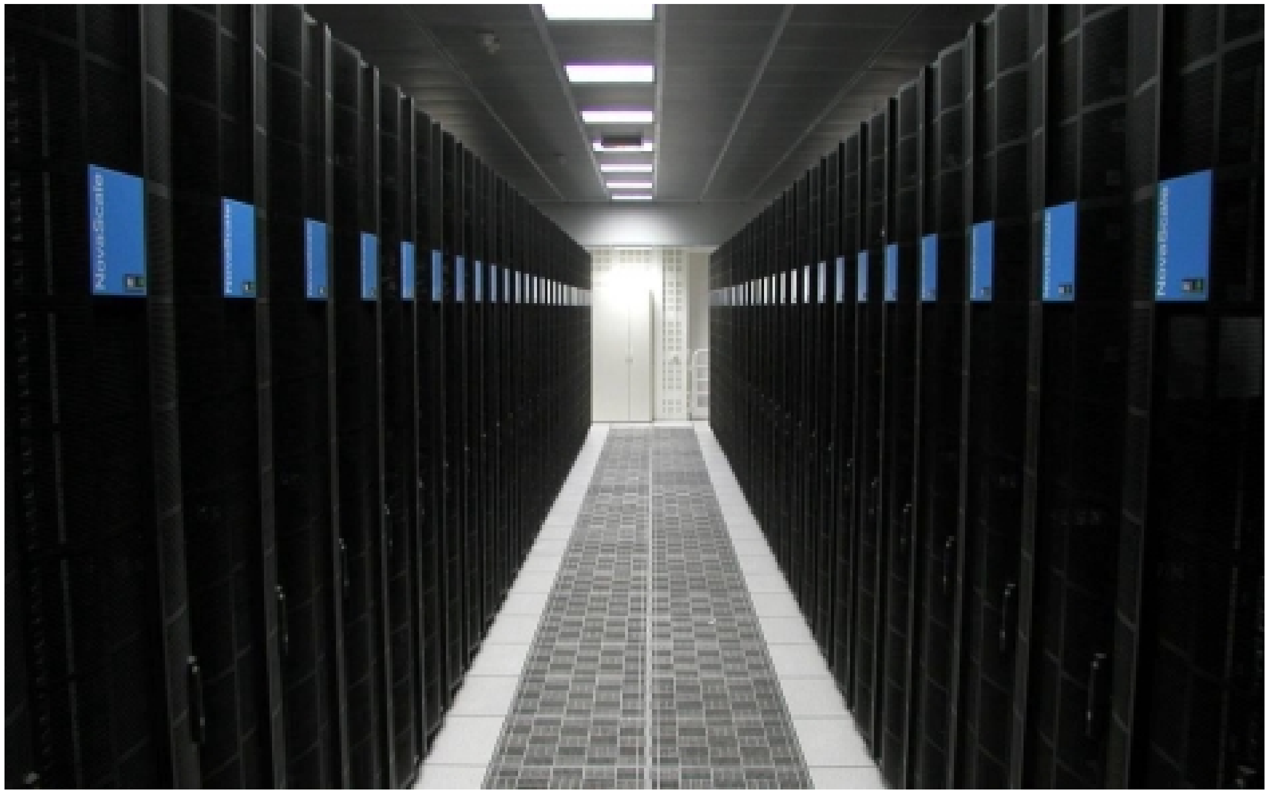
coarsening : LLNL, blue: SGSJ, red dashed: IC, red: ICp, green: SAINVp



“elapsed” time (s) for 3D Poisson as a function of p

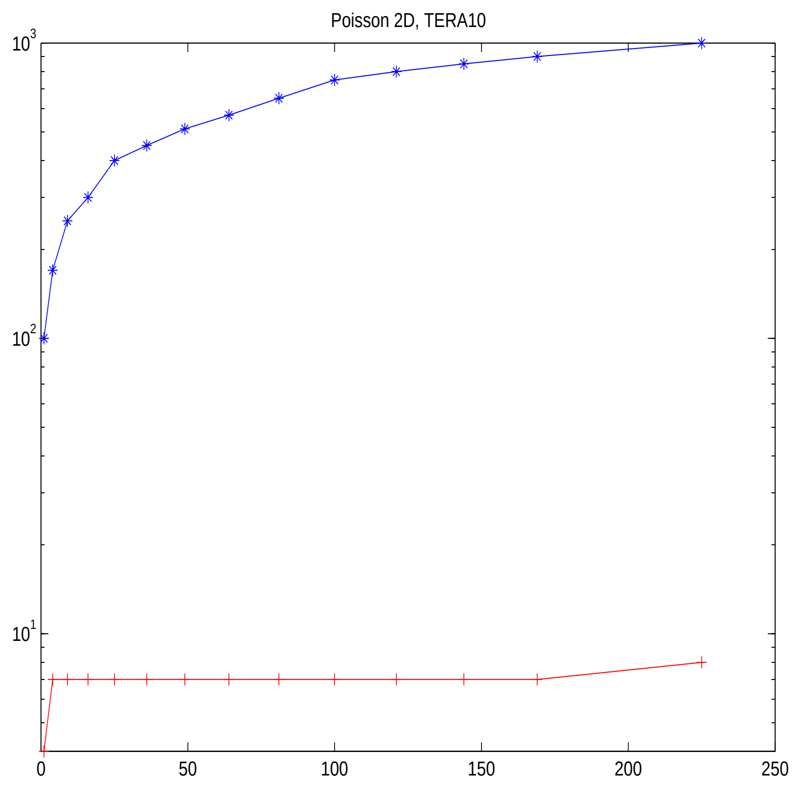
coarsening : LLNL, blue: SGSJ, red dashed: IC, red: ICp, green: SAINVp

The CEA TERA 10 parallel computer



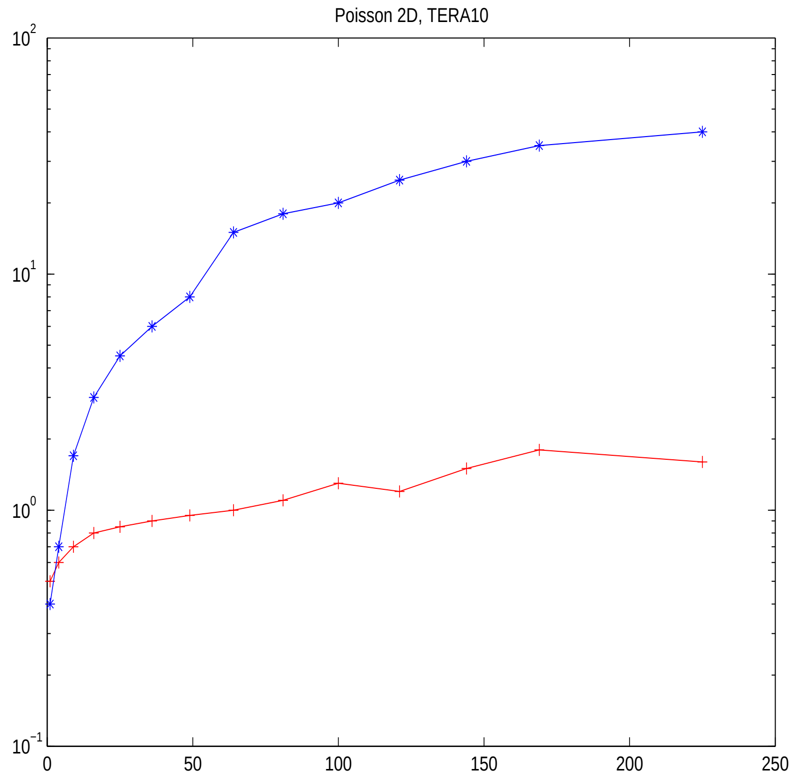
NOVASCALE
cea
TERA 10
Bull

	TERA-1	TERA-10
Processor	Alpha EV6 - 1 Ghz	Intel Montecito - 1.6 Ghz
Node	4 processors	16 cores (8 Montecito)
Memory per node	4 GB - 16 GB - 32 GB	48 GB - 128 GB
Peak performance	8 Gflops	> 100 Gflops
Interconnexion Network	2 "rails" ELAN - 3 Latency 5 us - Links 400 MB/s	3 "rails" ELAN - 4 Latency 4 us - Links 900 MB/s
Number of nodes	608	544
Peak performance	5 Tflops	> 60 Tflops
Sustained performance	1.35 Tflops	12.5 Tflops
Memory size	3 TB	30 TB
Disk space	50 TB	1 PB
Disk bandwidth	7.5 GB/s	100 GB/s
Storage network	32 HiPPI links (800 Mbits/s)	20 IB 4x links (1 GB/s)
User access	20 1 Gbits links	10 10 Gbits links



Nb of iterations for 2D Poisson as a function of p

TERA10, red: AMG, blue: IC, $10\ 000$ unkn/p



Elapsed time (s) for 2D Poisson as a function of p

TERA10, red: AMG, blue: IC, $10\ 000$ unkn/p

Extension to block matrices

Goal: solve linear systems arising from PDE systems (several unknowns per element or node)

Example: 3 temperature radiative transfer model

$$\rho \frac{\partial E_i(T_i)}{\partial t} = \mathsf{div}(K_i \nabla T_i) + \alpha(T_e - T_i),$$

$$\rho \frac{\partial E_e(T_e)}{\partial t} = \mathsf{div}(K_e \nabla T_e) - \alpha(T_e - T_i) - c(a\sigma_E T_e^4 - \sigma_A T_r^4),$$

$$\rho \frac{\partial a T_r^4}{\partial t} = \mathsf{div}(K_r \nabla a T_r^4) + c(a\sigma_E T_e^4 - \sigma_A T_r^4).$$

Unknowns are T_e, T_i, T_r (ρ is known)

This is a system of nonlinear PDEs whose behavior depends on the relative values of diffusion and relaxation terms

Using finite volumes, one obtains a nonlinear system with $3 \times N$ unknowns which is linearized with the Newton's method

We can solve these systems with a (point) multilevel preconditioner

However, results are not always so good

This motivated the development of a block extension of the AMG preconditioner

In our example, blocks are 3×3

Smoothers

∘ iterations of (symmetric) block Gauss–Seidel/Jacobi

Small $p \times p$ systems are solved by Gaussian elimination

∘ block IC/ILU

Influence matrix

Define influences between blocks:

block I depends on block J (J influences I) if

$$\|A_{I,J}\|_F \geq \tau \max_{K \neq I} \|A_{I,K}\|_F$$

This gives S of order n/size of blocks

Coarsening

The (block) graph of A is coarsened using S with the same algorithms as in the point case

Interpolation P

Interpolation is done component by component:

$$e_I = \sum_{J \in C_I} W_{I,J} e_J,$$

C_I: coarse nodes influencing I, $W_{I,J}$ diagonal $p \times p$ matrix, no coupling between different types of unknowns

Use the same formula as in the point case

$$R = P^T$$

$$A_{\text{grossier}} = RAP$$

This couples the unknowns

Model problem

Block 5-diagonal symmetric matrix with 3×3 blocks

Constant diffusion and relaxation coefficients

Diagonal blocks:

$$\begin{pmatrix} 4\alpha + \mu h^2 & -\mu h^2 & 0 \\ -\mu h^2 & 4\alpha + (\mu + \sigma) h^2 & -\sigma h^2 \\ 0 & -\sigma h^2 & 4\alpha + \sigma h^2 \end{pmatrix}$$

Nonzero nondiagonal blocks are $-\alpha I_3$, $h = 1/(m+1)$

$\alpha = 1, \mu = 10, \sigma = 200$

m	n	nb it block GS	nb it block IC
10	300	6	5
20	1200	7	6
30	2700	7	6
40	4800	7	6
50	7500	7	6

Conclusions

- These multilvel preconditioners are (almost) scalable
- Drawback: setup phase is “expensive”
- They are useful only for difficult and/or very large problems