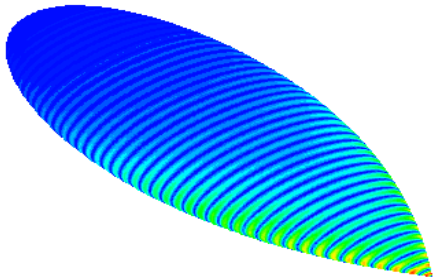
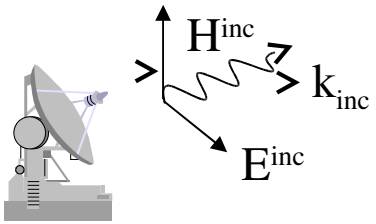
High Performance Methods to solve large 3D Electromagnetic Problems

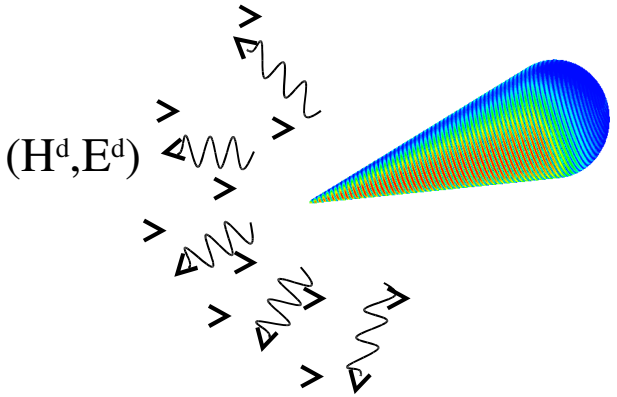


David GOUDIN
CEA/DAM/CESTA

What is the problem ?

The simulation of the electromagnetic behavior for a complex target in the frequency domain formulation





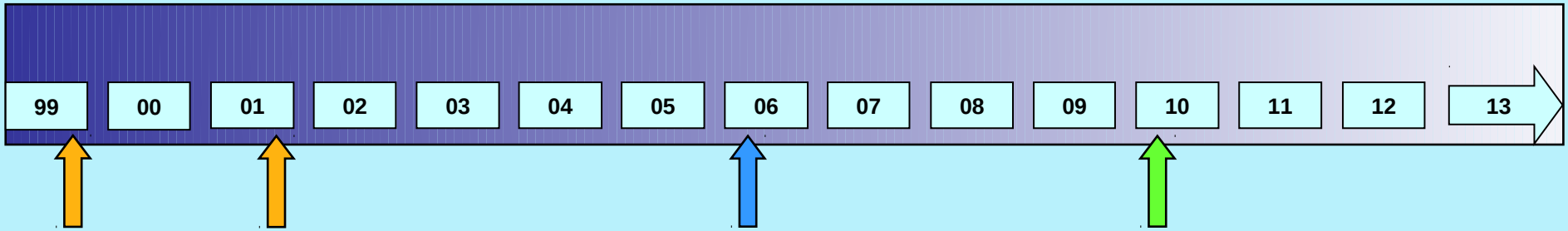
⇨ Numerical solution of Maxwell's equations in the free space

- ❑ RCS (Radar Cross Section) computation
- ❑ Antenna computation

$$RCS = 10\log_{10}(\sigma) \qquad \sigma = \lim_{r \to +\infty} 4\pi r^2 \frac{\left|\vec{E}^d(r)\right|^2}{\left|\vec{E}^{inc}(r)\right|^2}$$

Our Supercomputers

TERA



30 (50) Gflops

TERA-1 : 1 (5) Tflops

TERA-10 : >10 Tfops

TERA-100 : >100Tflops

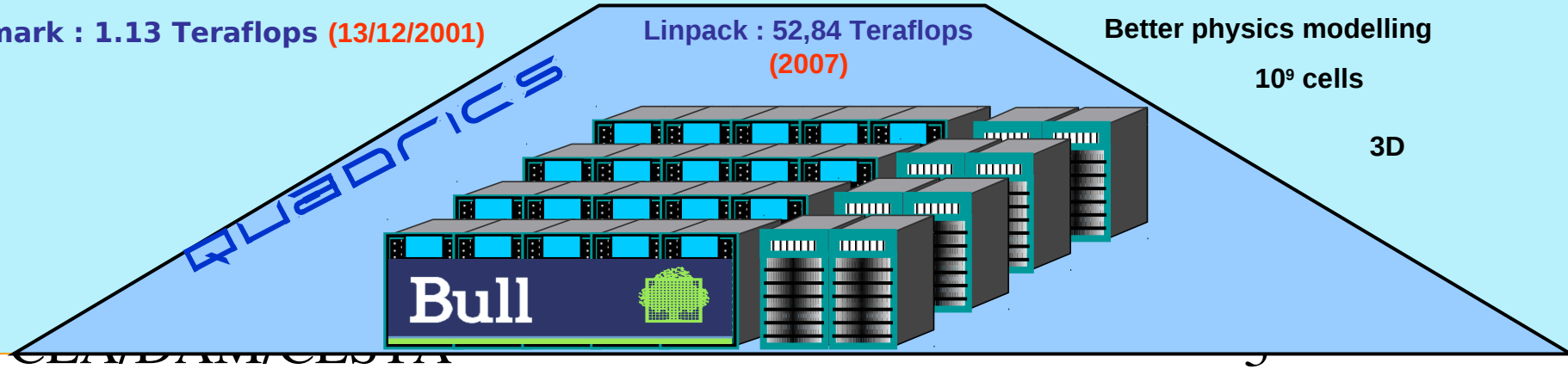
CEA benchmark : 1.13 Teraflops (13/12/2001)

Linpack : 52,84 Teraflops (2007)

Better physics modelling

10^9 cells

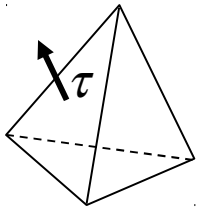
3D



Partial Differential Equations Methods = PDE

- volumic description
- the unknowns : fields in the computational volume : *E* and/or *H*
- formulations
  - ➔ *2nd order on E*
  - ➔ *2nd order on H*

⇨ Discretization by volumic finite elements H(rot)



Boundary Integral Equations Methods = EI

- surfacic description
- the unknowns : equivalent currents on the surface : *J* et *M*
- formulations
  - ➔ *EFIE*
  - ➔ *CFIE*
  - ➔ *EID (CEA originality)*

⇨ Integral representation :

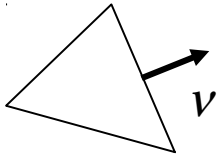
$$E^d = i\omega\mu L_\omega(J) - \frac{1}{i\omega\varepsilon} grad\,(L_\omega(div_\Gamma J)) - ro$$

$$H^d = i\omega\varepsilon L_\omega(M) - \frac{1}{i\omega\mu} grad\,(L_\omega(div_\Gamma M)) + i$$

¿

¿ ¿ ¿

⇨ Discretization by surfacic finite elements H(div)



❑ **Fully BIEM**

❑ **Meshes at the surface and on interfaces between homogeneous isotropic medium**

- number of DOF (Degrees of Freedom) reasonable
- leads to a full matrix

$$\begin{pmatrix} A^s_{22} & A^s_{23} \\ A^s_{23} & A^s_{33} \end{pmatrix} \begin{pmatrix} M \\ J \end{pmatrix}$$

⇨ Parallelization : *very efficient*

❑ **Solver**

In house parallel Cholesky-Crout solver.
The matrix is :

- symmetric
- complex
- non hermitian

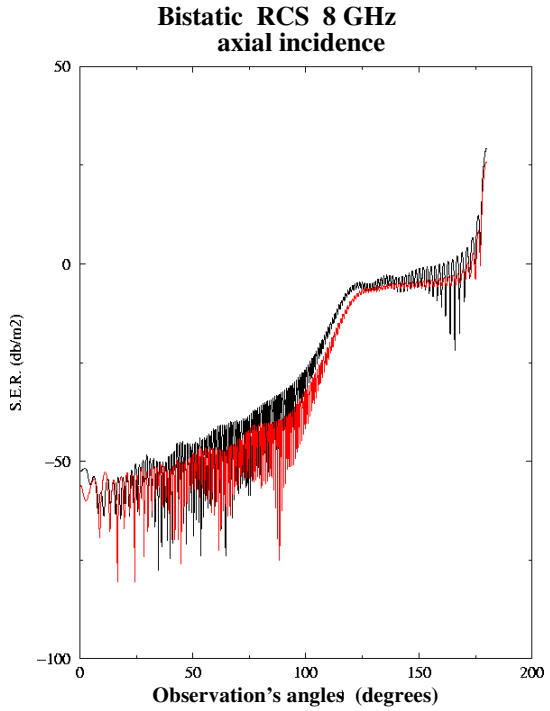
For some applications, the matrices are Non symmetric so we use

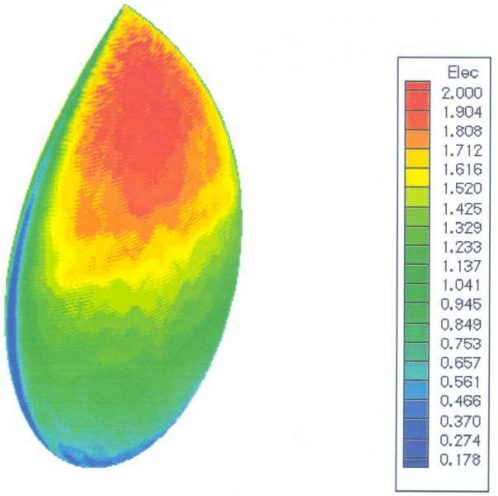
- The Scalapack LU solver

- **NASA Almond** *8 Ghz*

  233 027 unknowns with 2 symmetries ⇒ 932 108 unknowns

  - matrix size : 434 GBytes
  - 4 factorizations & 8 back/forward solves to compute the currents
  - global CPU time on 1536 processors : **36 083 seconds**





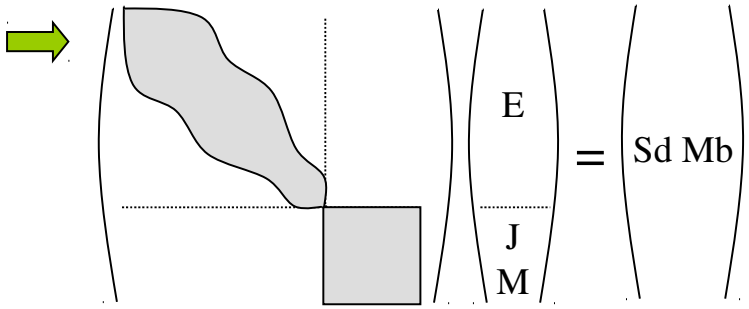
The limits of the full BIEM method

BIEM

⊖

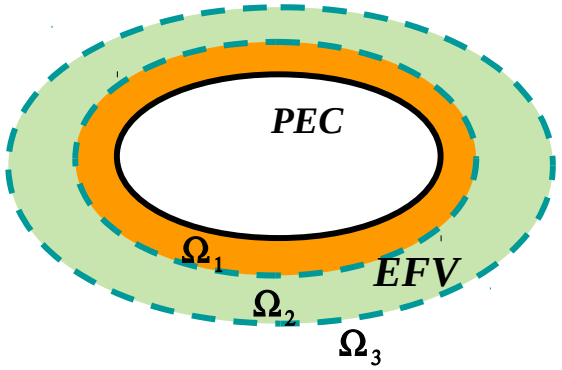
- Only homogeneous and isotropic media
- Numerical instability for very thin layers
- Size of the dense linear system ↗ according to the number of layers

A solution : a strong coupling PDE - BIEM



- It needs to use a Schur complement : elimination of the unknown E
- **BUT need of a great number of computations (solutions) of the sparse linear system**

The other solution: a weak coupling PDE - BIEM



Based on

* a domain decomposition method (DDM) partitioned into concentric sub-domains
* a Robin-type transmission condition on the interfaces
* on the outer boundary, the radiation condition is taken into account using a new IE formulation called EID
* we solve this problem with inner/outer iteration

① Gauss-Seidel for the global problem

Inside each sub-domain

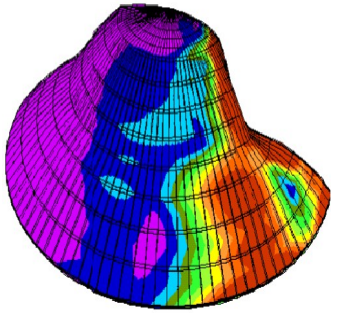
- ② iterative solver (// conjugate gradient)
- the PaStiX direct solver (EMILIO library)

for the free space problem

- ② a // Fast Multipole Method
- a direct Scalapack solver

EMILIO in few words

- EMILIO was developed in collaboration with INRIA's team
- EMILIO uses efficient parallel implementation of direct methods to solve sparse linear systems, thanks to the high performance direct solver PaStiX and the Scotch package (both developed by INRIA's team)
- Organized into two parts :
  - a sequential pre-processing phase using a global strategy,
  - a parallel phase.



- In our 3D code, we use an old version of PaStiX
  - 1-D block column distribution
  - Full MPI implementation
  - Static scheduling

Sparse Direct Solver on NUMA and Multicore Architectures

Xavier Lacoste, Mathieu Faverge, Pierre RAMET

Solstice Workshop, Toulouse, France

INRIA Bordeaux - Sud-Ouest

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

June 16, 2010

Introduction

PaStiX solver (http://pastix.gforge.inria.fr)

- Bacchus Project (INRIA Bordeaux - Sud-Ouest)
- Sparse direct solver
- Supernodal method
- Hybrid implementation: MPI for distributed memory, PThread for shared memory
- Static scheduling
- 1D-block column distribution or 2D-block distribution

Goals

- Develop a dynamic scheduler adapted to multi-core architectures
- Implement Schur computation
- Improve OOC efficiency

PaStiX Functions

- LLt, LDLt, LU factorization with supernodal implementation
- Static pivoting + Refinement: CG/GMRES
- 1D/2D block distribution + Full BLAS3
- Simple/Double precision + Float/Complexe operations

- MPI/Threads implementation (SMP/Cluster/Multicore)
- Sequential or distributed interface
- Support external ordering library (PT-Scotch/METIS)

- Require only C + MPI + Posix Thread
- Multiple RHS (direct factorization)
- Incomplete factorization ILU(k) preconditionner
- Out-of Core implementation (in SMP mode)

PaStiX Release

- Latest version: "shakespeare" (04/10 - svn 2985/2995)

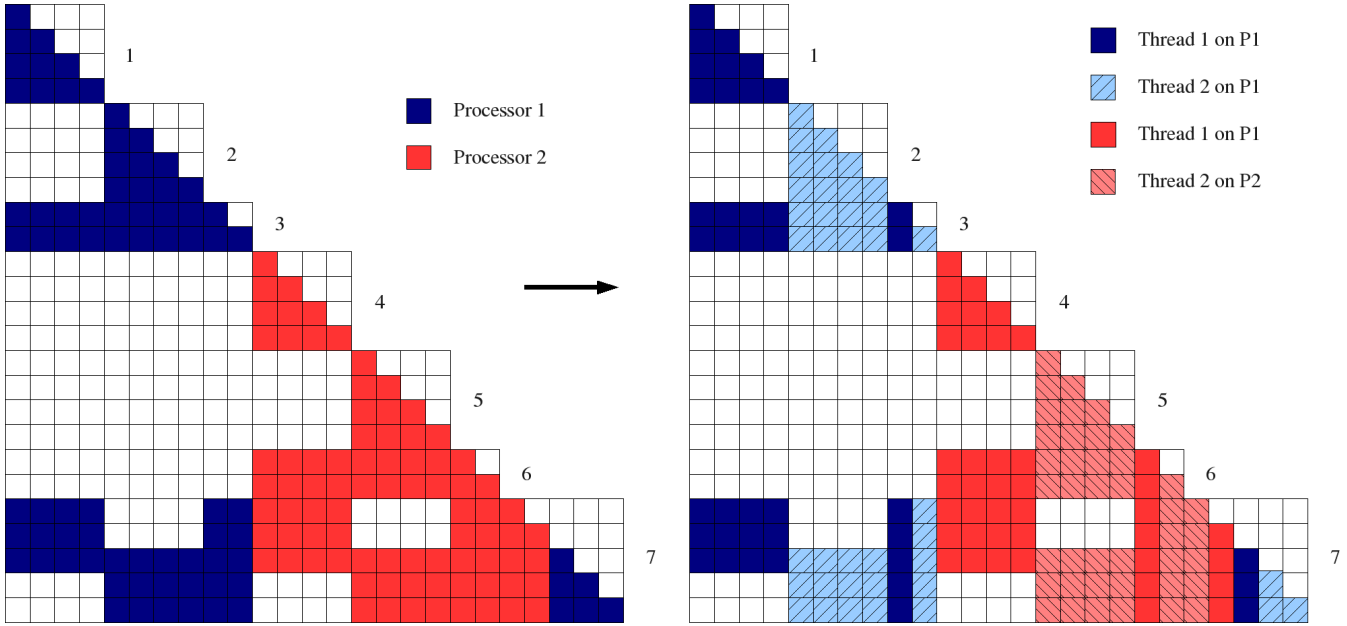
Highlights

The direct solver PaStiX has been successfully used by CEA/CESTA to solve a huge symmetric complex sparse linear system arising from a 3D electromagnetism code on the TERA-10 CEA supercomputer.

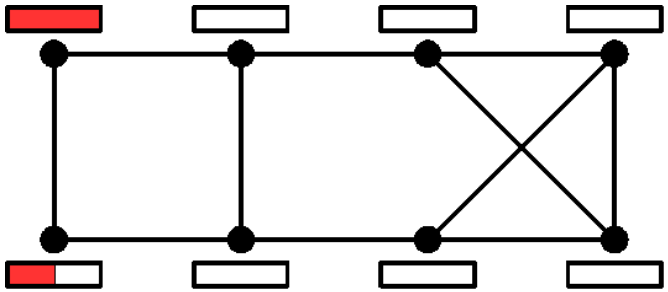
- **45 millions unknowns**: required 1.4 Petaflops and was completed in half an hour on 2048 processors.
- **83 millions unknowns**: required 5 Petaflops and was completed in 5 hours on 768 processors.

To our knowledge a system of this size and this kind has never been solved by a direct solver.

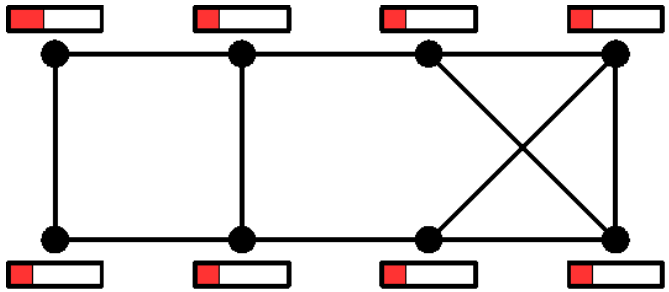
NUMA-Aware Allocation (upto 20% efficiency)



(a) Localization of new NUMA-aware allocation in the matrix



(b) Initial allocation



(c) New NUMA-aware allocation

Dynamic Scheduling

Needs

- Adapt to NUMA architectures
- Improve memory affinity (take care of memory hierarchy)
- Reduce idle-times due to I/O (communications and disk access in a future works)
- Use dedicated threads for communications and disk access

Proposed solution

- Based on a classical work stealing algorithm
- Steal is limited to preserve memory affinity
- Use dedicated threads for I/O and communications in order to give them an higher priority

Study on a large test case: 10M

Properties

N	10 423 737
NNZ_A	89 072 871
NNZ_L	6 724 303 039
OPC	4.41834e+13

	4x32	8x32
Static Scheduler	289	195
Dynamic Scheduler	240	184

Table: Factorization time in seconds

- Electromagnetism problem in double complex from CEA Cesta
- Available on GridTLSE platform
- Cluster Vargas from IDRIS with 32 power6 per node

Conclusion and Future Works

Conclusion

- NUMA-aware allocation gives good results (about 20%)
- Results with dynamic scheduling show improvements on SMP and NUMA architectures (about 10%)
- Dynamic scheduling with ESP method (about 15%)

Future Works

- Study an hybrid approach mixing a 2D-block distribution and the dynamic ESP method
- Schur complement : centralized implementation, promising results with CEA Cesta benchmarks.
- OOC : still not working with EDF benchmarks...

PaStiX: `http://pastix.gforge.inria.fr`

MURGE: `http://murge.gforge.inria.fr`

Gantt diagram are done with ViTE software now available in release 1.0 on `http://vite.gforge.inria.fr`

Other softwares developed in our group:
Scotch: `http://scotch.gforge.inria.fr`
Hips: `http://hips.gforge.inria.fr`

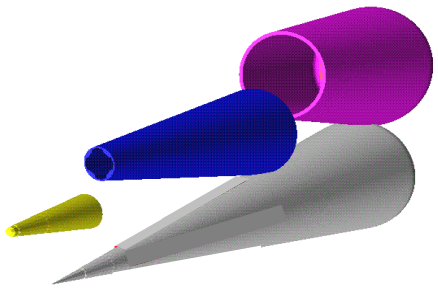
MURGE : a common API to the sparse linear solvers

`http://murge.gforge.inria.fr`

Features (inside HIPS & PaStiX)

- Through one interface, one can access to many solver (direct/iterative) strategies.
- One can enter a graph/matrix in a centralized/distributed way.
- Simple formats : coordinate, CSR or CSC.
- Very easy to implement an assembly phase using MURGE.
- MURGE proposes Fortran and C prototypes.

How we use PaStiX (Direct Solver) in EMILIO



Sequential Computation (1 time per problem)

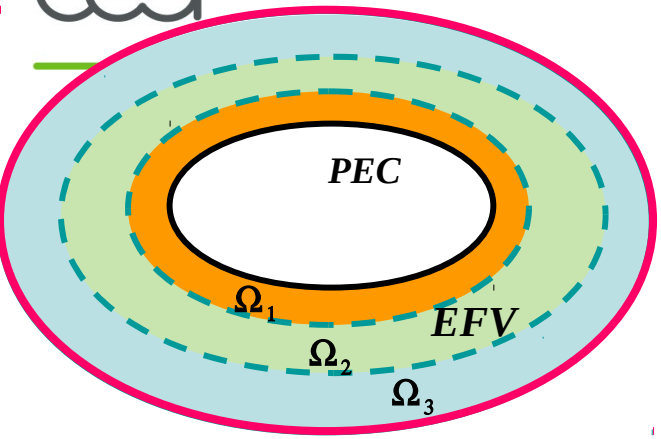
Scotch

Reordering of the unknowns

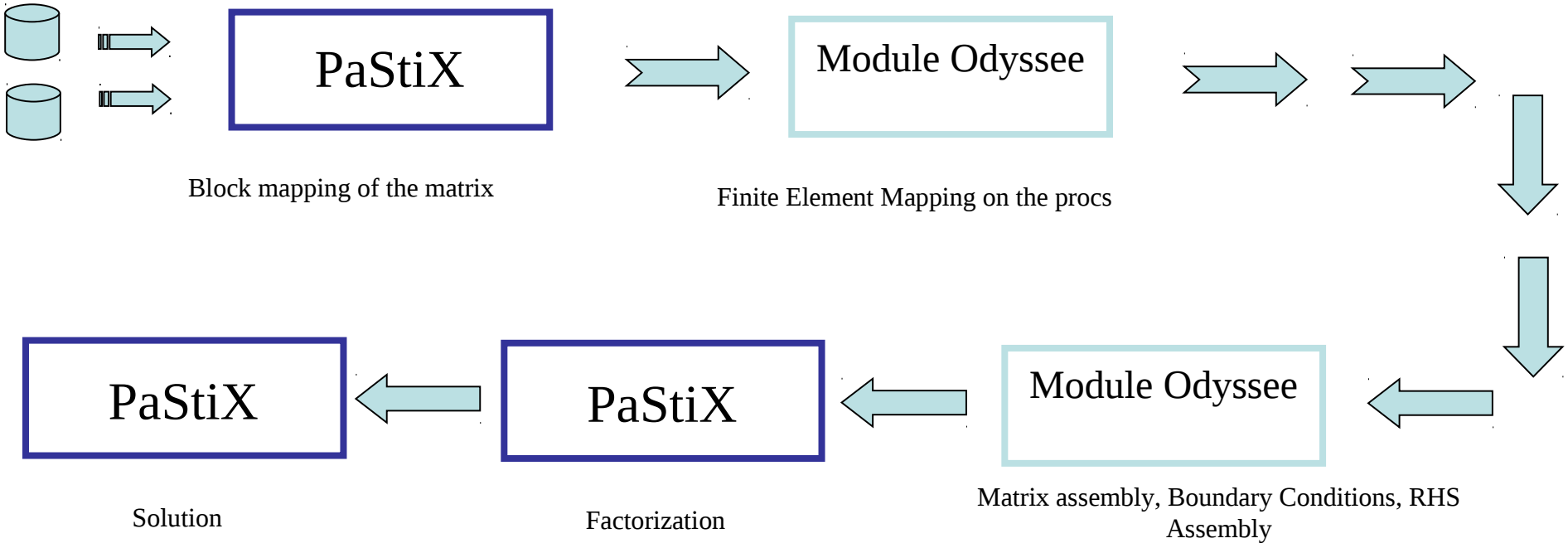
PaStiX

Symbolic Factorization

How we use PaStiX (Direct Solver) in EMILIO

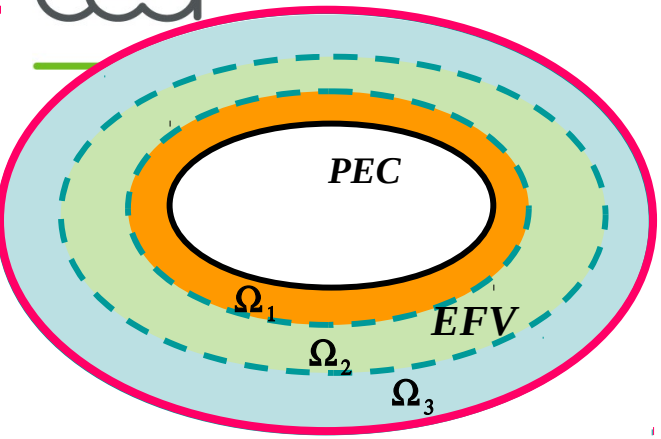


Parallel Computation for the 1st iteration for each volumic subdomain



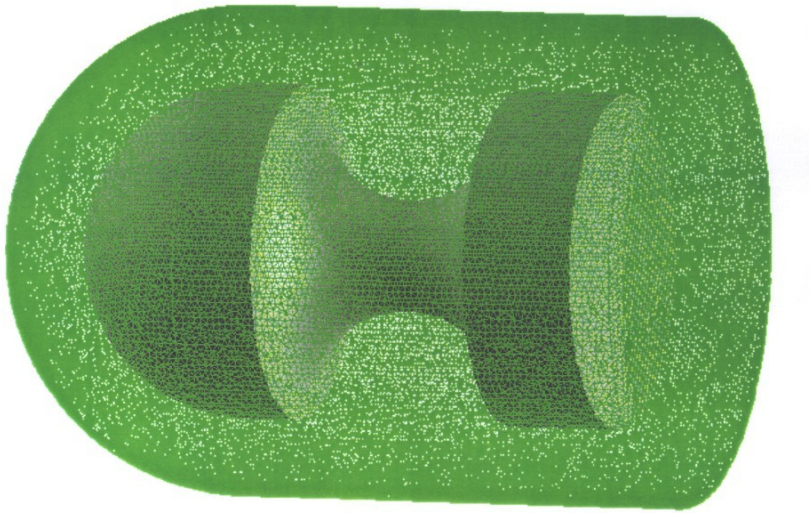
How we use PaStiX (Direct Solver) in EMILIO

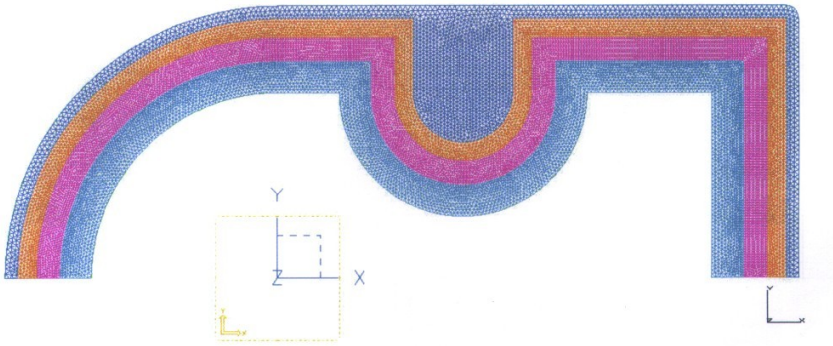
BIEM : EID



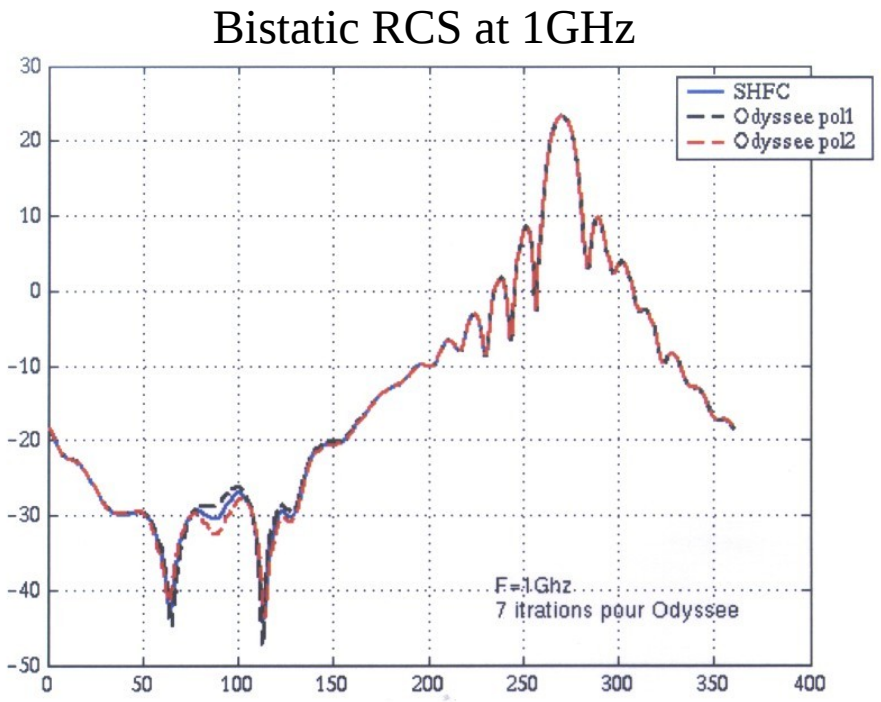
Parallel Computation for the iteration > 1 for each volumic subdomain

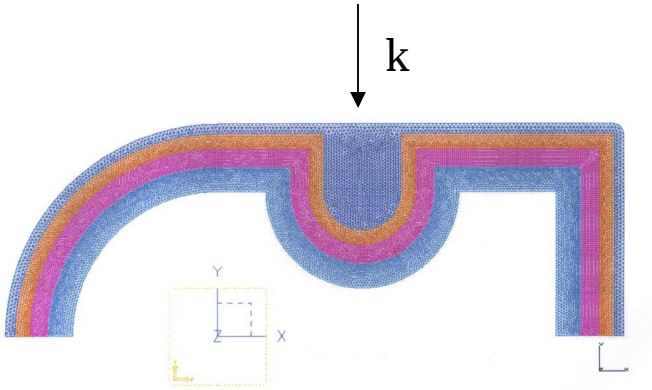
- Altere :
  - * **5.63 millions** unknowns in 4 sub-domains
  - * **120 000** edges for the outer boundary





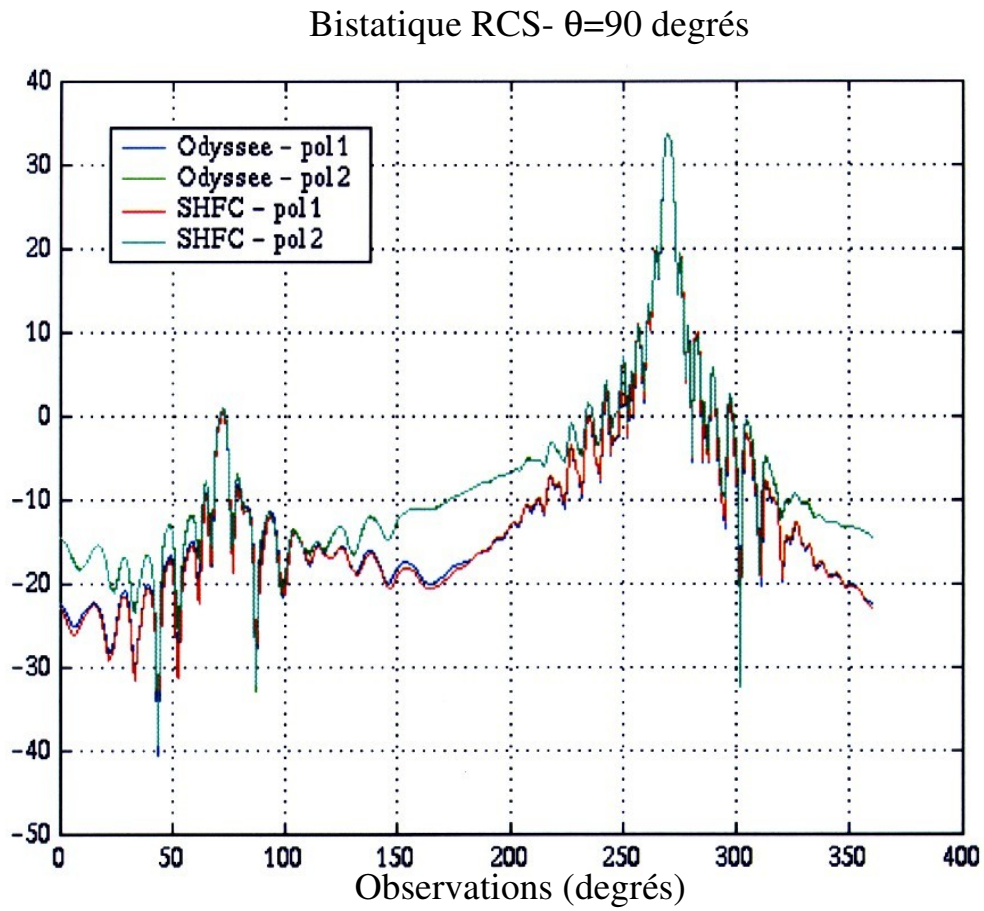
- Comparison 2D axi-symmetric / Odyssée



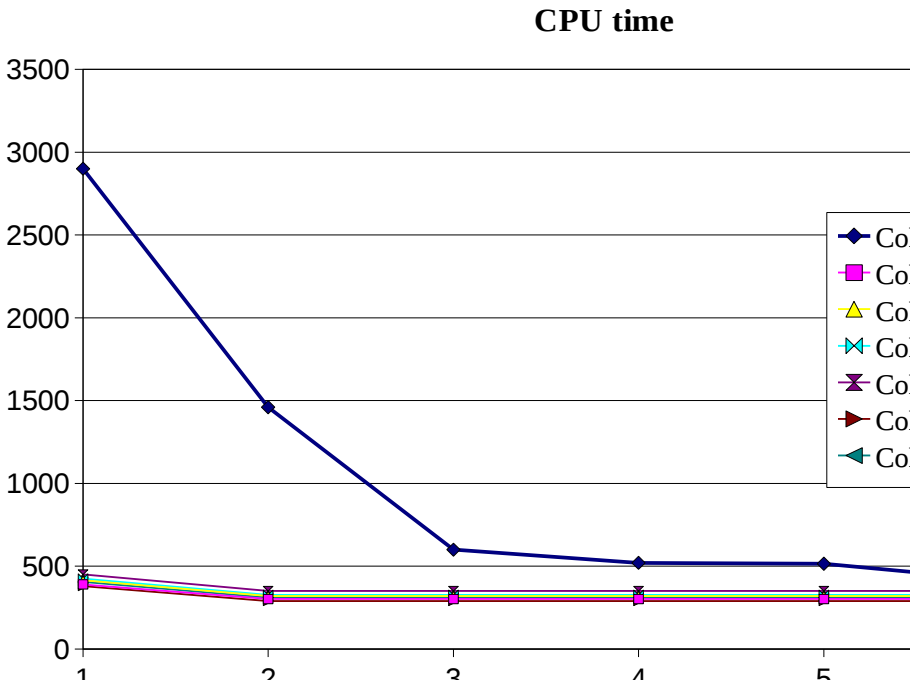


Numerical results for an another test

Comparison 2D axi-symmetric / Odyssee

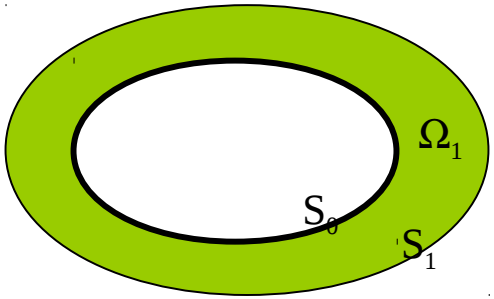


- 7 iterations for the relaxed Gauss-Seidel



Our limit : 23 millions of unknowns

- A complex object :
  - * **23 millions** of unknowns in **1 sub-domain**
  - * **20 000** unknowns for the EID



Ω_1 : EMILIO – full MPI
(industrial version of PasTiX)

S_1 : EID + MLFMA

About 125 Tera operations needed for the factorization only

Number of procs	512 (32 nodes)
Number of MPI tasks	64 (2 per node)
Memory used per Node	94 Go
Time to Assembly the Matrix	870 s
Time of Factorization	8712 s
Time of Resolution	40 s
Time for the EID	30 s

How to bypass the gap of Memory consumption

Problem : Find a high performance linear solver able to solve very large sparse linear systems (hundreds of millions of unknowns)

Existing software :

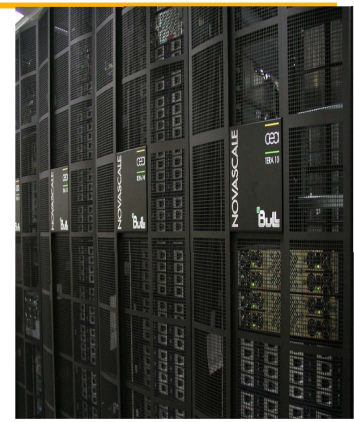
- EMILIO (with solver PaStiX full MPI) already integrated in ODYSSEE
- PETSc (iterative solver) already integrated in ODYSSEE

Limits :

- Emilio is limited to 23 Millions of unknowns
- Classical Iterative solver are not well adapt to our problems

- Find New solvers
- Use the PTScotch software to avoid the ordering sequential step
- Test the MPI+Threads version of PaStiX (results on the next slide…)

The biggest problem (for the moment with TERA10…) solved by PaStiX MPI+Threads



➡ 3D Problem : 82 Millions of unknowns
(New PasStiX MPI + Threads)

OPC LDLt : 4.28 e+15
NNZ in A : 5.97 e+10

Number of cores (Tasks MPI / Threads)	Min Time Factorization per core	Max Time Factorization per core	Min Memory per SMP Node	Max Memory per SMP Node
768 (48/16)	**4703 s**	**27750 s**	**58 Go**	**115 Go**

Other Softwares studied to bypass the GAP !

Iterative and Multigrid Methods (Joint work with D. Lecas)

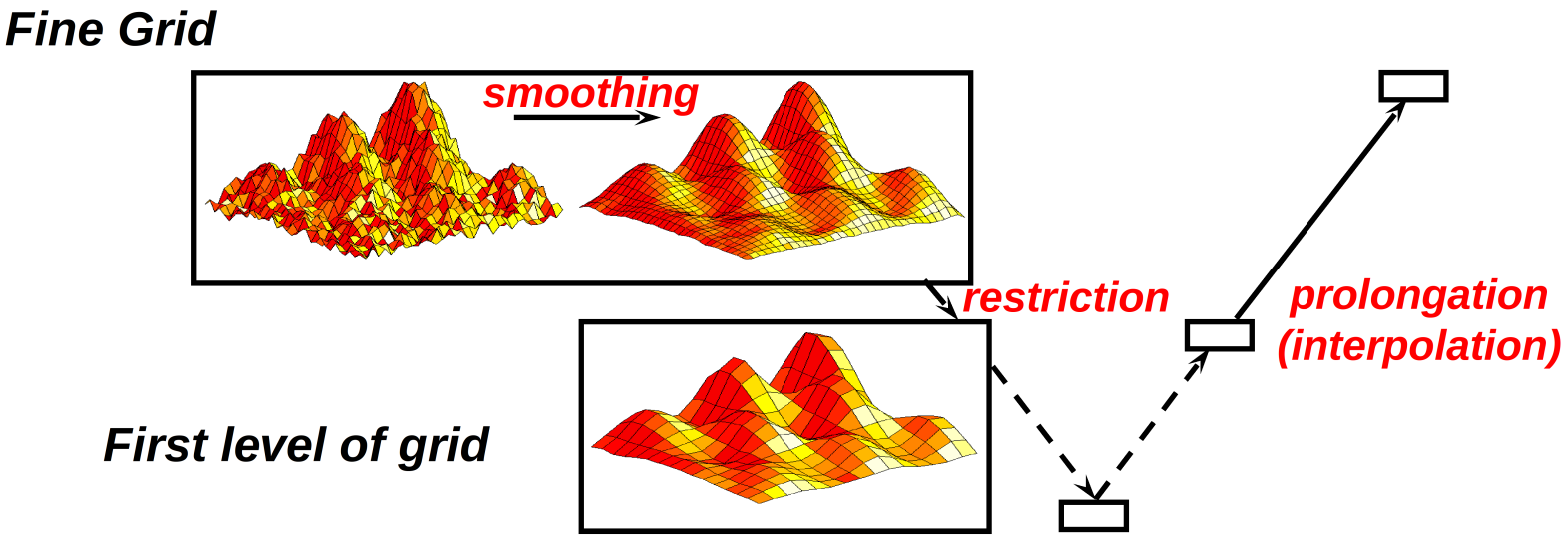
- ILUPACK
- Hypre
- PETSc
- HIPS : Hierarchical Iterative Parallel Solver, solver developped by the INRIA team Bacchus
- MaPHYS : developped by the INRIA team Hiepacs

Direct Methods

- SuperLU
- MUMPS

A parallel hybrid multigrid/direct method

- Compute solution using direct solver on initial mesh
- Prolongate solution on finer mesh
- Apply smoother and compute residual
- Restrict residual on coarse mesh and compute error e
- Apply correction using prolonged e then smooth
- Go to Step 2 until desired refinement level is reached



Conclusions for Parallel codes for electromagnetism

- We have developed a 3D code which is able to take into account all the goals we want to attain
- We successfully validated all the physics through comparison with :
  - other codes we have (full BIEM, 2D axis symmetric)
  - measurements

Future research for Odyssee

- We must finish to develop a complete hybrid, implementation of Odyssee with MPI + Threads
- Improve our methods and our softwares, test some iterative methods for the solution of each sub domain in the volume,…
- Include some softwares developped during the ANR (need more time…)