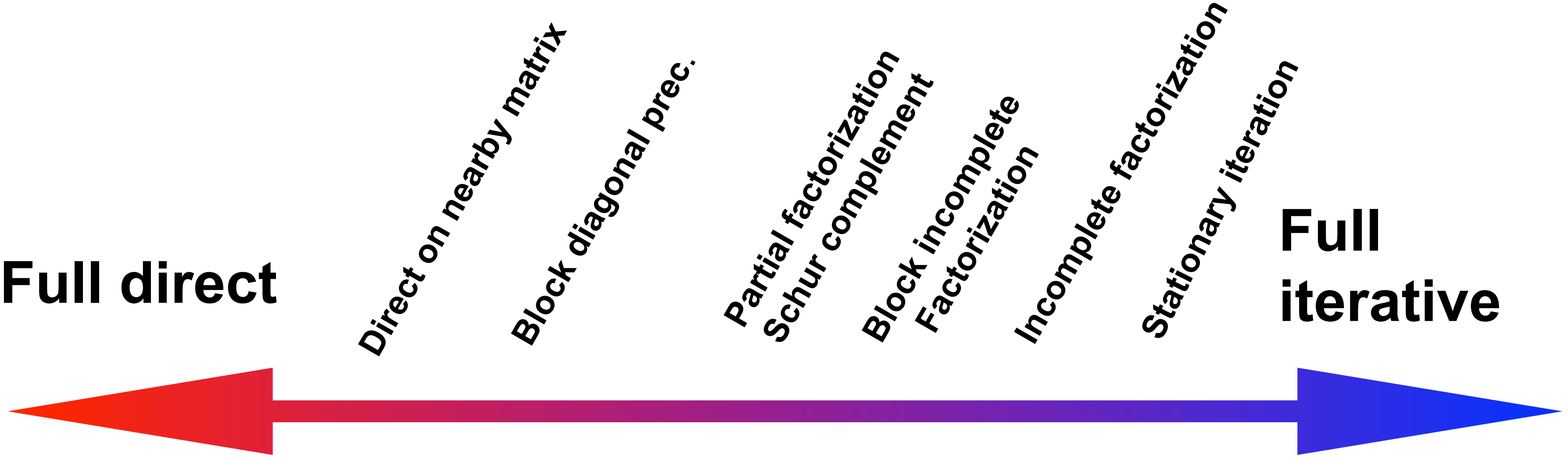
Parallel Numerical Linear Algebra

Paris, Jan. 13, 2015

Numerical Linear Algebra Solver

Goal: solve Ax = b, where A is large and sparse



Full direct

Direct on nearby matrix

Block diagonal prec.

Partial factorization
Schur complement

Block incomplete
Factorization

Incomplete factorization

Stationary iteration

Full iterative

Usual trades of

Direct

- Robust/prescribed accuracy for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large3D problem
- Limited weak scalability

Iterative

•Problem dependent efficiency/ monitored accuracy

•Sparse computational kernels

•Less memory requirements and possibly faster

•Potential high weak scalability

Parallel Implementation – Task Based Paradigm

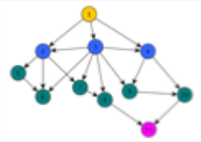
Linear algebra

$$AX = B$$

Sequential-Task-Flow

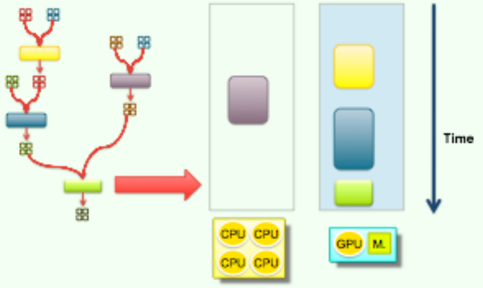
```
for (j = 0; j < N; j++)
  Task (A[j]);
```

Direct Acyclic Graph



→

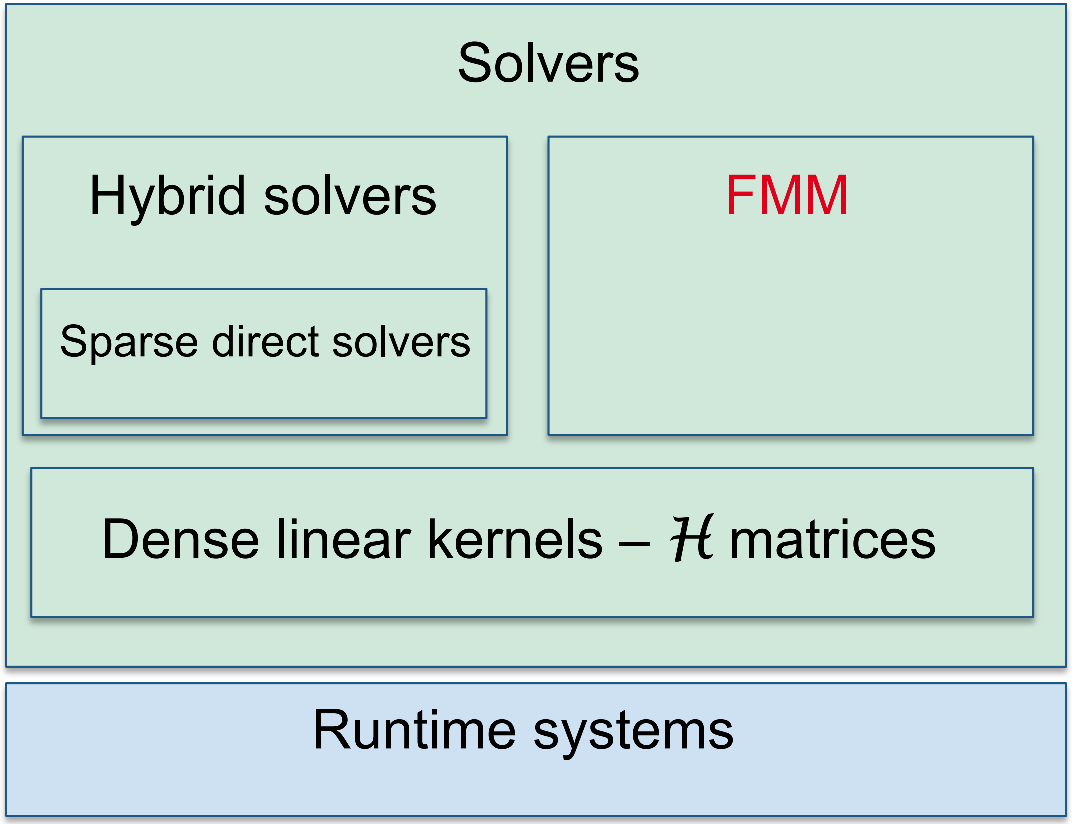
Runtime systems



→

Heterogeneous platforms

Foreseen software stack outcome



Solvers

Hybrid solvers

Sparse direct solvers

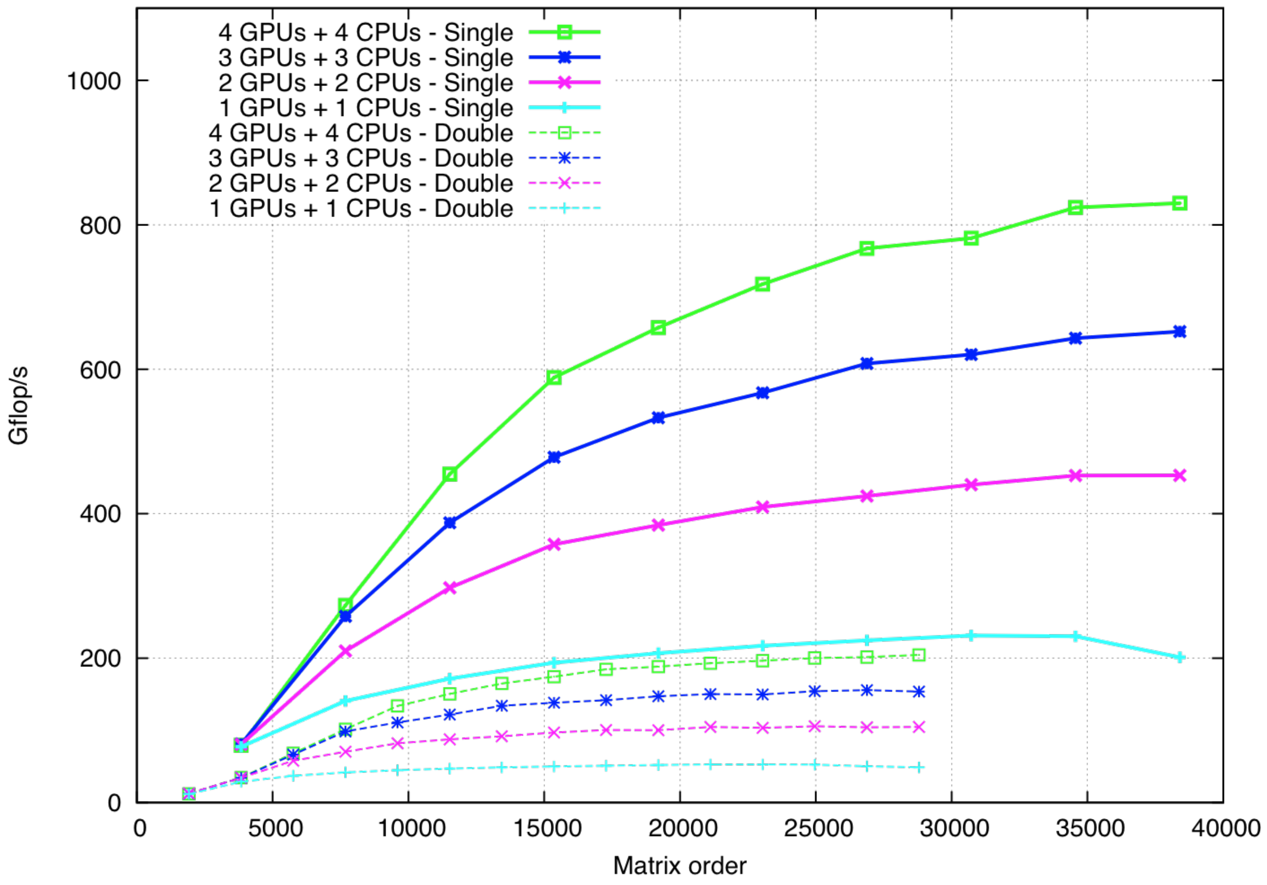
FMM

Dense linear kernels – $\mathcal{H}$ matrices

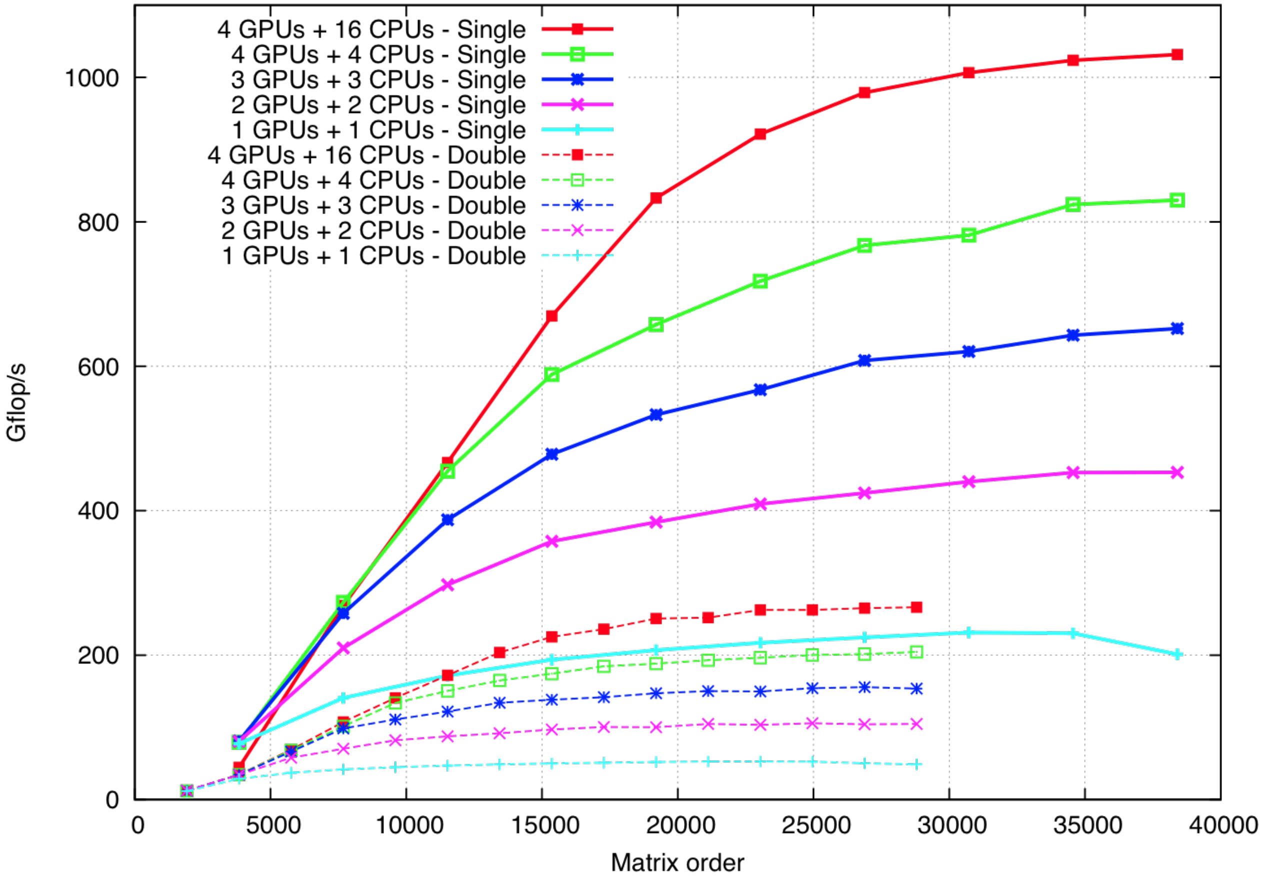
Runtime systems

All the stack on parallel sparse linear algebra components

Dense linear algebra : QR factorization



Dense linear algebra : QR factorization



+ 200 Gflop/s but 12 cores = 150 Gflop/s

Dense Linear Algebra

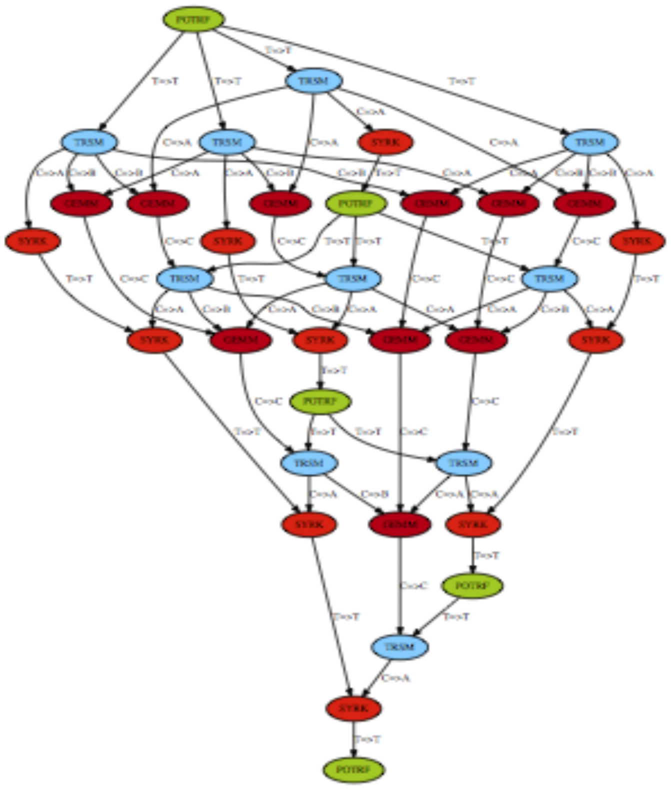
- Ongoing activities/related projects
  - Distributed implementation
  - Application driven scheduling
  - 2 PhD thesis within an Conseil Régional –Inria project

    (CEA, BSO: HiePACS, RealOpt, Runtime)
  - ANR Solhar
  - AE MORSE (HiePACS, Runtime, ICL, UCD, KAUST)

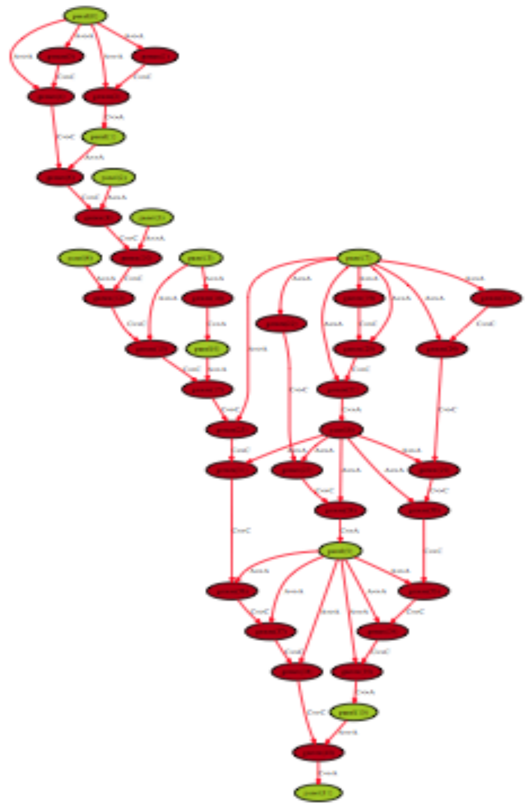
Sparse Direct Solver- PaStiX

- Numerical features
  - LLT , LDLT , LU factorization with supernodal implementation
  - Static pivoting + Refinement: CG/GMRES
  - 1D/2D block distribution + Full BLAS3
  - Simple/Double precision + Float/Complex operations
- Implementation features
  - MPI/Threads implementation (SMP/Cluster/Multicore/ NUMA)
  - Dynamic scheduling inside SMP nodes (static mapping)
  - Support external ordering library (PT-Scotch/METIS)

Sparse direct solver- PaStiX

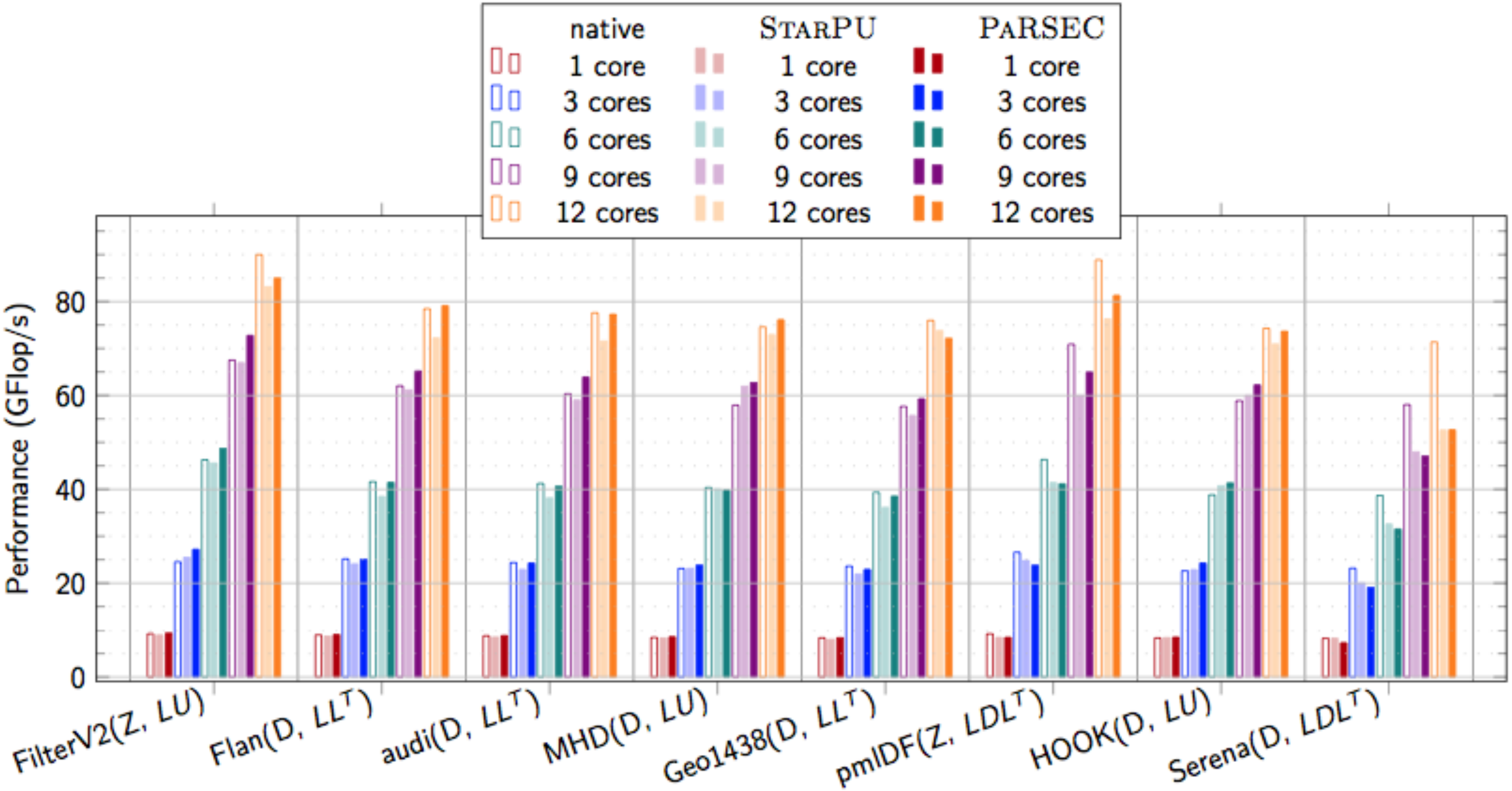


(c) Dense DAG

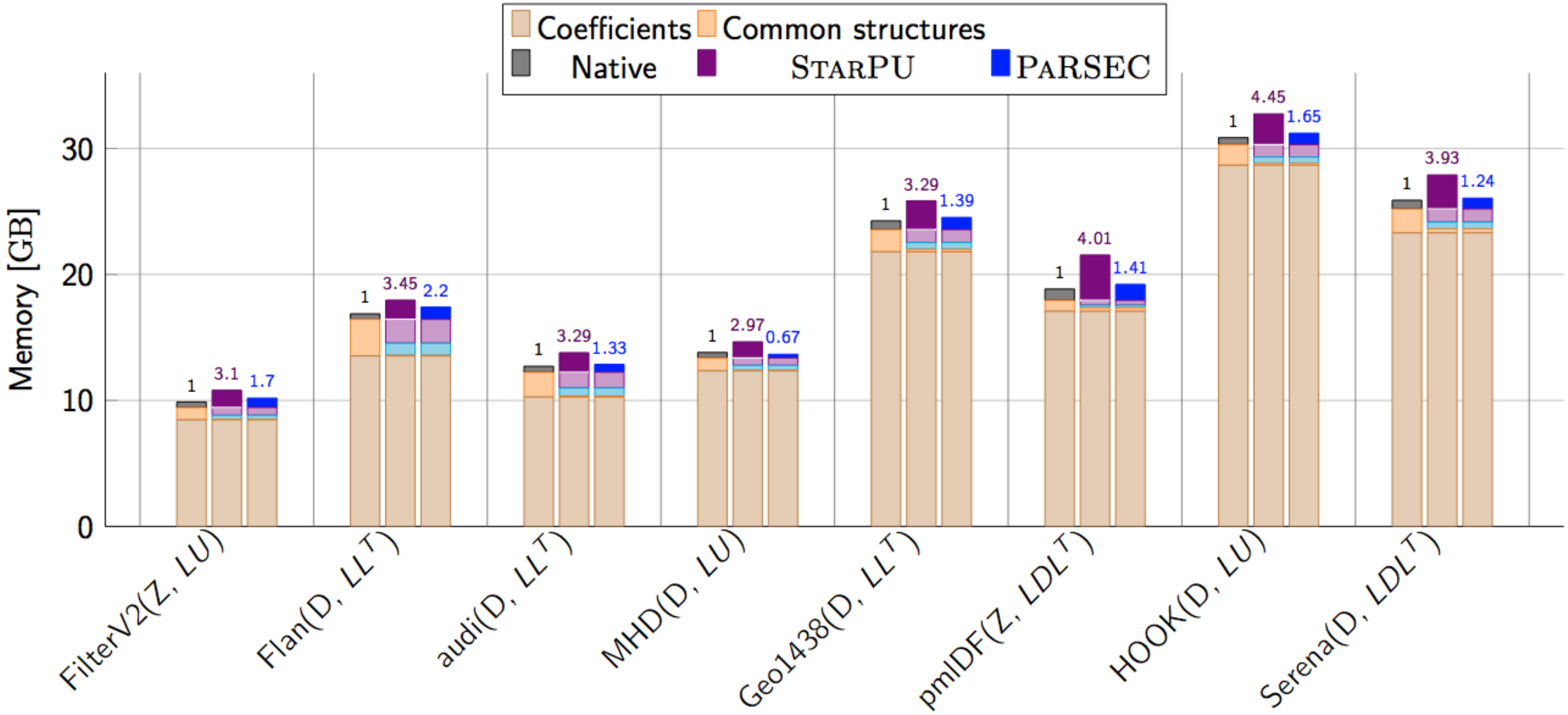


(d) Sparse DAG representation of a sparse LDL^T factorization

Sparse direct solver- PaStiX performance



Sparse direct solver- Memory overhead



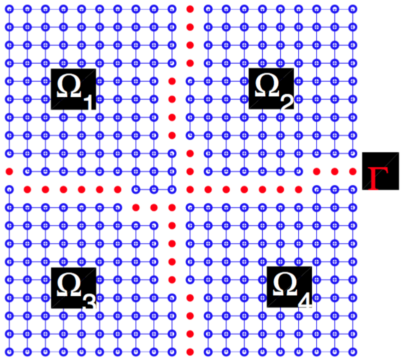
PhD funded by ANR ANEMOS

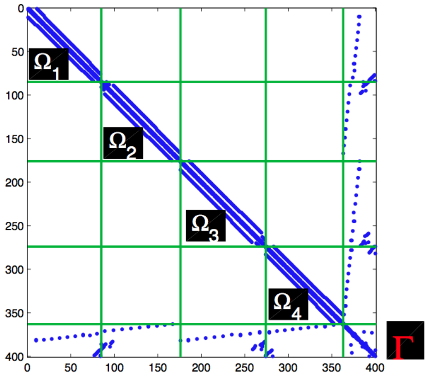
Ongoing related projects

- ANR ANEMOS, Solhar
- HPC-PME (AlgoTech) follow-up Fortissimo

Hybrid Direct-Iterative Solver: MaPHyS

- Partition the adjacency graph of the sparse matrix (Scotch)





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

- Local calculation of Schur complements and precondtioner

(MUMPS, PaStiX)

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

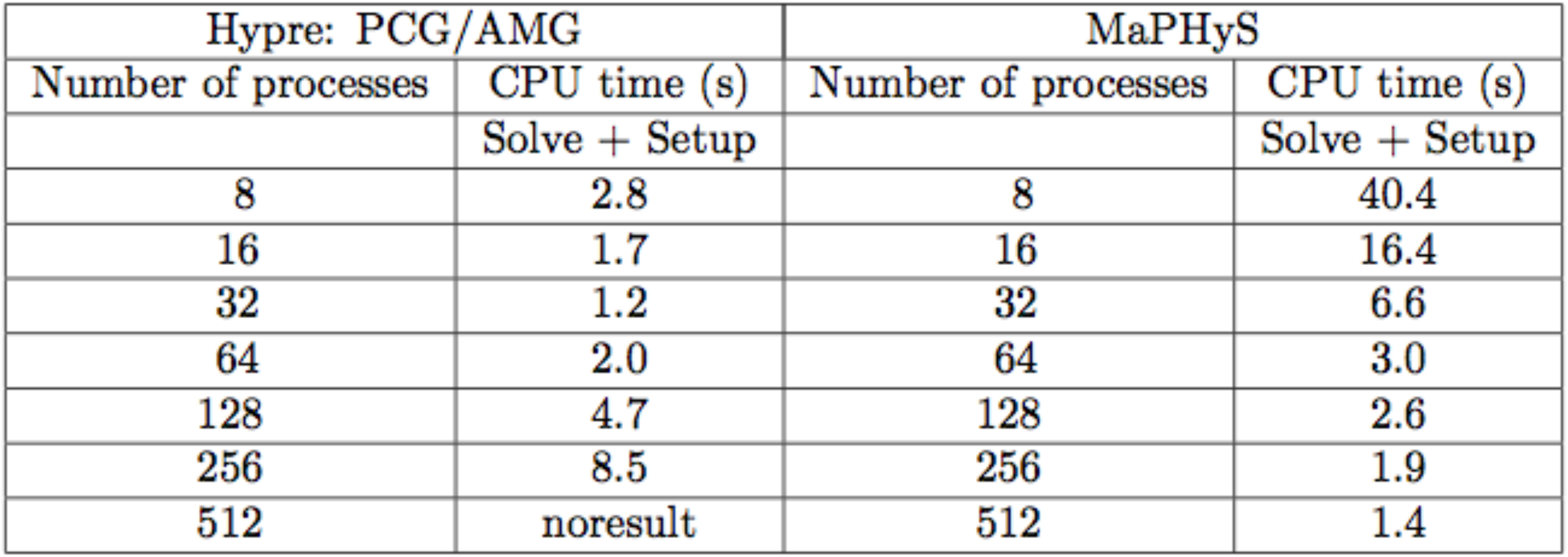
(Magma, MUMPS, PasTiX)

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

- Parallel hierachical implementation: MaPHyS

Hybrid direct-iterative solver: MaPHyS

ANDRA test problem: 1.3 MDof

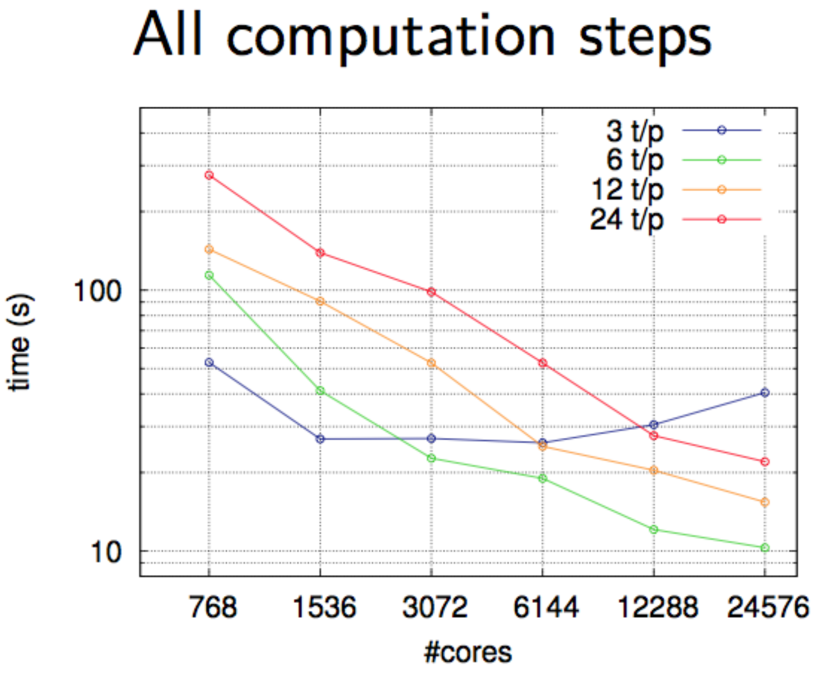


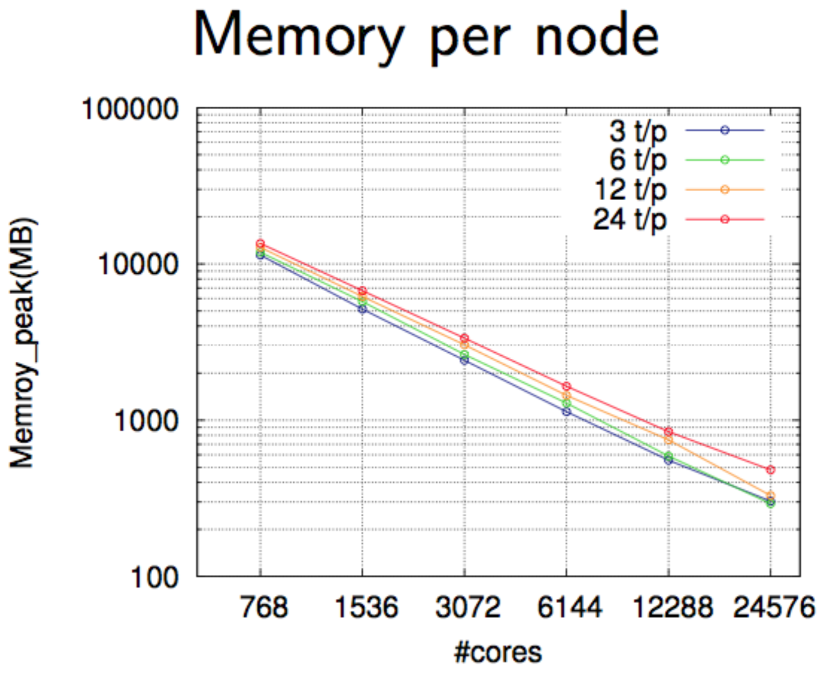
Hypre: PCG/AMG		MaPHyS	
Number of processes	CPU time (s)	Number of processes	CPU time (s)
	Solve + Setup		Solve + Setup
8	2.8	8	40.4
16	1.7	16	16.4
32	1.2	32	6.6
64	2.0	64	3.0
128	4.7	128	2.6
256	8.5	256	1.9
512	noresult	512	1.4

Preliminary results – full MPI

Hybrid Implementation – Hybrid solver: MaPHyS

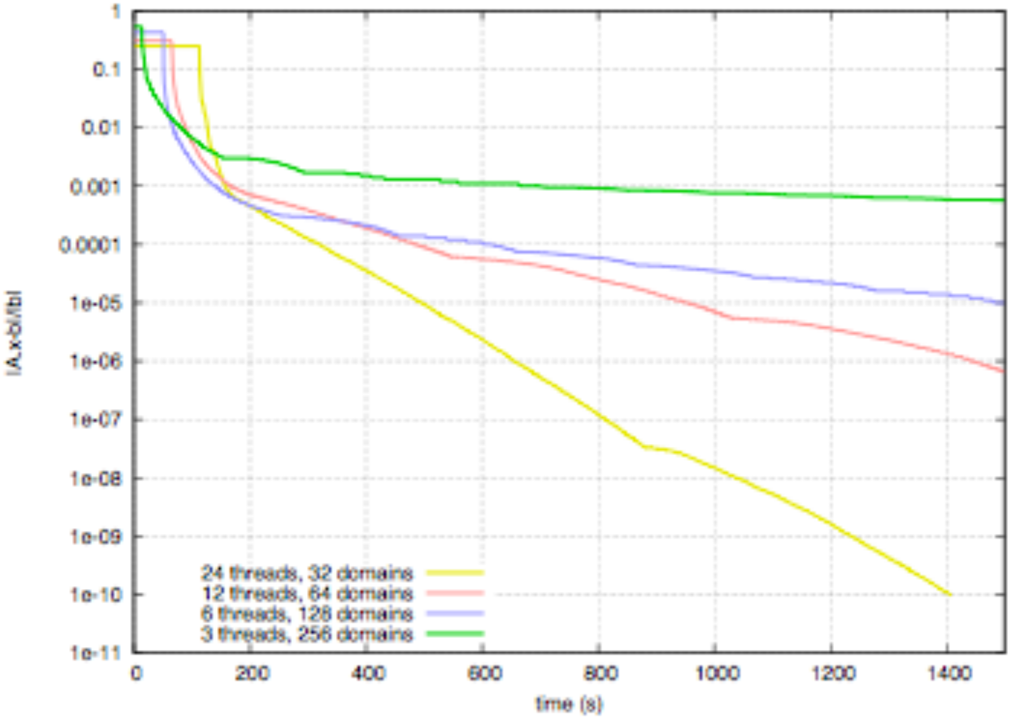
NACHOS test problem: 4 Mdof

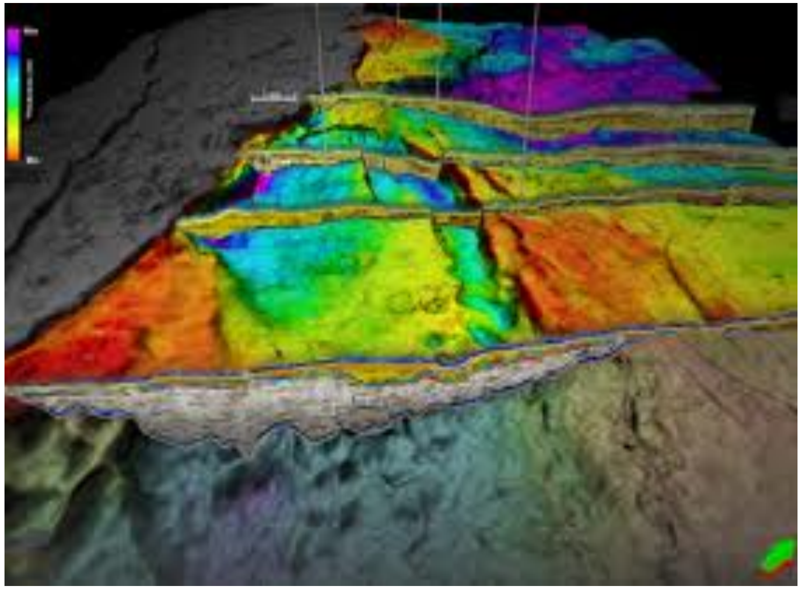




Hybrid direct-iterative solver: MaPHyS

MAGIQUE3D test problem: 1.2 Mdof





Statoil modeling

PhD funded by Total, DIP framework

IJD funded by IPL C2S@Exa, follow-up by Labex CPU

Ongoing related projects

- ANR DEDALES – POMDAPI
- ANR TECSER - NACHOS

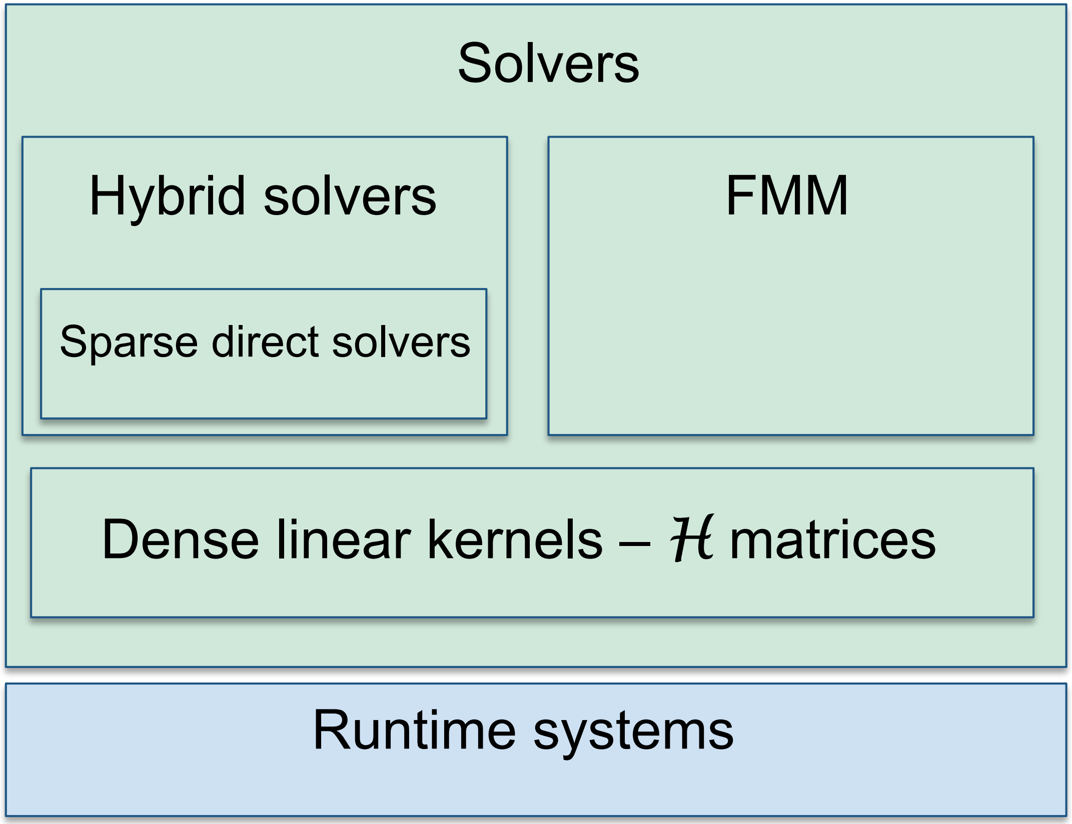
Krylov solver for extreme scale

- FP7 Exa2CT project (Alpines, HiePACS, SAGE)
  - ✓ Résilience hard-soft error
  - ✓ s-step/block methods using hiding communication numerical schemes
  - ✓ Communication avoiding ILU preconditioner

- Related ongoing projects:

  ANR TECSER, Rapid Hi-Box

Foreseen software stack outcome


Solvers

Hybrid solvers

Sparse direct solvers

FMM

Dense linear kernels – $\mathcal{H}$ matrices

Runtime systems

All the stack on parallel sparse linear algebra components