High performance numerical algorithms and tools

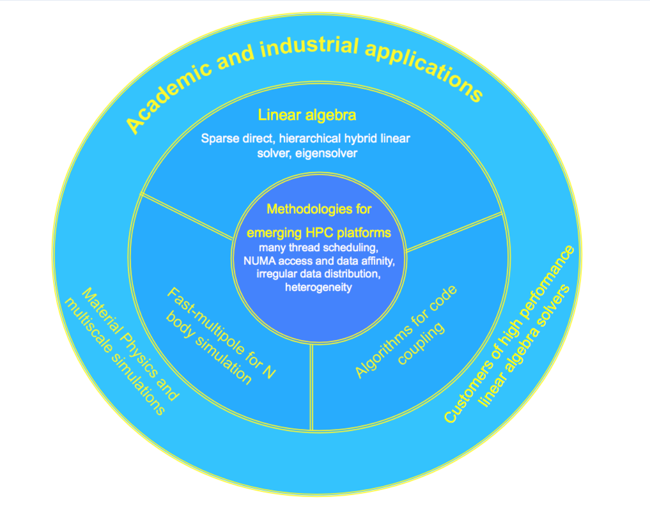
HiePACS - Inria Bordeaux Sud-Ouest

First Brazil-France workshop
Petropolis, September 17-19, 2012

HiePACS onion scientific structure

Objectives: contribute to the effort towards Peta/Exascale computing

A multidisciplinary approach



Outline

1. Fast Multipole Method on a Runtime System

2. Parallel sparse hybrid linear solver

3. Parallel geometric multigrid

4. Concluding remarks

Outline

1. Fast Multipole Method on a Runtime System
2. Parallel sparse hybrid linear solver
3. Parallel geometric multigrid
4. Concluding remarks

Fast Multipole

Joint work with (FAST-LA associated team)

Éric DARVE (Stanford University)
Toru TAKAHASHI (Nagoya University)

Fast Multipole Background (1/3)

Objectives

Compute

$$f_i = \sum_{j=1}^{N} P(x_i, y_j) w_j \quad \text{for } i = 1, \ldots M$$

P is the dense matrix (kernel of a Green's function, that decays with distance).

Main decomposition idea

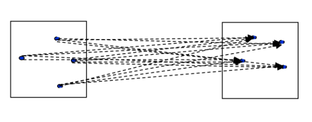
$$f_i = f_i^{near} + f_i^{far}$$

where f_i^{far} approximated by analytical expansions

1. Development: Taylor, spherical harmonic
2. Interpolation (black-box) [Darve & Fong, JCP, 2009]

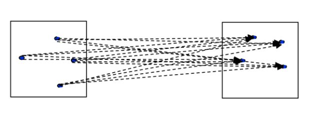
Fast Multipole Method (2/3)

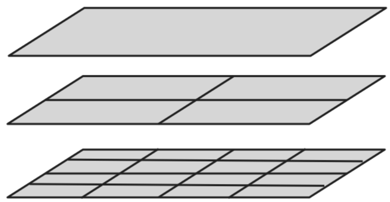
Far field calculation: a shortcut

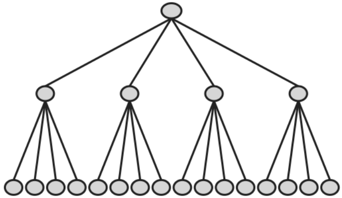


Fast Multipole Method (2/3)

Far field calculation: a shortcut

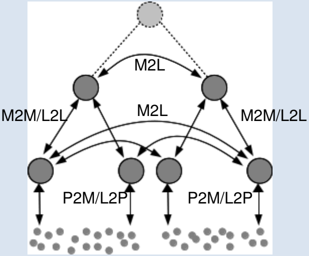






Fast Multipole Method (3/3)

Simplified FMM tree

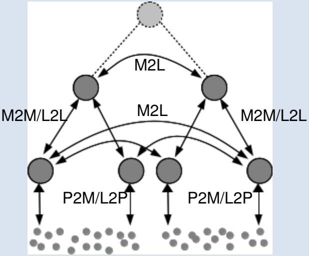


FMM main operators

- P2P : Particule to Particule (near field)
- P2M : Particule to Multipole (up)
- M2M : Multipole to Multipole (up)
- M2L : Multipole to Local
- L2L : Local to Local (down)
- L2P : Local to Particule (down)

Fast Multipole Method (3/3)

Simplified FMM tree



FMM main operators

- P2P : Particule to Particule (near field)
- P2M : Particule to Multipole (up)
- M2M : Multipole to Multipole (up)
- M2L : Multipole to Local
- L2L : Local to Local (down)
- L2P : Local to Particule (down)

Implementation : three-layer paradigm

High-level algorithm

FMM

Runtime System

STARPU

Device kernels

CPU core
All 6 kernels

- P2P (optimized)
- M2L (non optimized)

Implementation : three-layer paradigm

High-level algorithm

FMM

Runtime System

STARPU

Device kernels

CPU core

All 6 kernels

GPU

- P2P (optimized)
- M2L (non optimized)

Implementation : three-layer paradigm

High-level algorithm

FMM

Runtime System

StarPU

Device kernels

CPU core

All 6 kernels

GPU

- P2P (optimized)
- M2L (non optimized)

Implementation : three-layer paradigm

High-level algorithm

FMM

Runtime System

STARPU

Device kernels

CPU core

All 6 kernels

GPU

- P2P (optimized)
- M2L (non optimized)

Implementation : three-layer paradigm

High-level algorithm

FMM

Runtime System

StarPU

Device kernels

CPU core

All 6 kernels

GPU

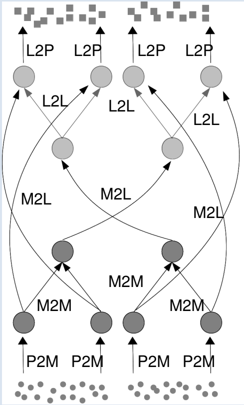
- ⋆ P2P (optimized)
- ⋆ M2L (non optimized)

StarPU runtime system

- Developped by Inria Runtime team;
- Ensures data coherency:
  - Modified Shared Invalid (MSI) protocol;
- Productive programming paradigm:
  - Implicit task dependency
- Unified runtime system
  - SMP/multicore, Nvidia GPUs, OpenCL devices, Cell processors;
  - Distributed memory;
  - Soon: Intel SCC and MIC.

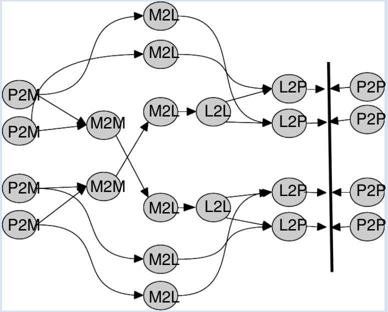
FMM task flow

Mirrored tree



- Vertex: data
- Edge: task

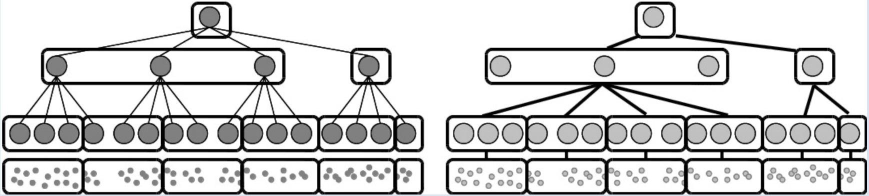
Task flow: far field | near field



- Vertex: task
- Edge: task dependency

Granularity

Gather leaves to coarsen the DAG



Set up

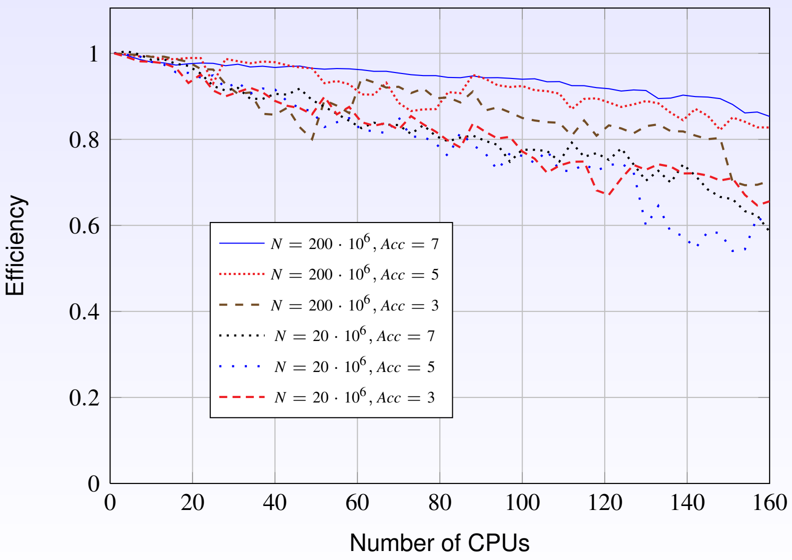
160 cores machine (SGI Altix UV 100)

- twenty octa-core Intel Xeon E7-8837 processors
- 2.67 GHz;
- ccNUMA;

Parallel Efficiency (e_p)

$$e_p = \frac{t_1}{p \times t_p},$$

160 cores machine - Performance



Nehalem-Fermi machine

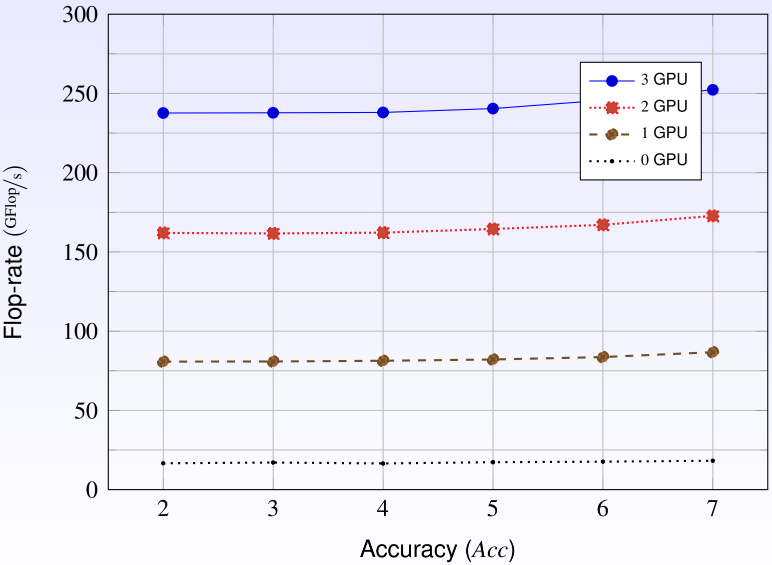
Host (12 cores)

- dual-socket hexa-core Intel X5650 Nehalem processors;
- 2.67 GHz;
- ccNUMA.

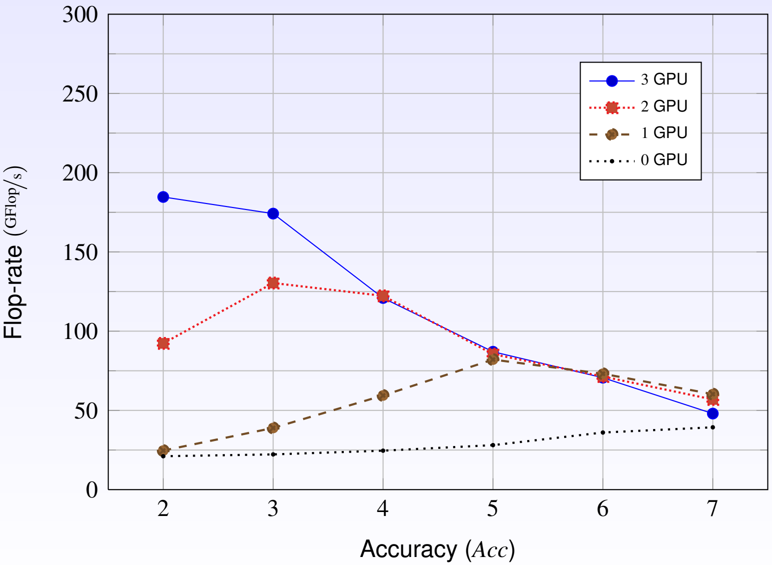
Devices (3 GPUs)

- Nvidia Fermi M2070
- 6 GB;
- 16x PCI bus to the host.

Non Uniform Distribution - Cube - h=7



Non Uniform Distribution - Cube - h=8



Conclusion

Conclusion

- ⋆ Performance portability across architectures;
- ⋆ Matrices Over Runtime Systems @ Exascale (MORSE):
  - Dense linear algebra (Magma);
  - Sparse direct solver (PaStiX);
  - FMM (ScalFMM) ;
- ⋆ Dominant far field:
  - Need to improve M2L kernel (on-going).

Future work

- ⋆ Distributed memory;
- ⋆ All 6 kernels on GPU;
- ⋆ AMD Tahiti (OpenCL) and Intel MIC accelerators;
- ⋆ DAGuE runtime system.

Outline

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- ★ Robust/prescribed accurate for general problems
- ★ BLAS-3 based implementations
- ★ Memory/CPU prohibitive for large $3D$ problems
- ★ Limited weak scalability

Iterative

- ★ Problem dependent efficiency / monitored accuracy
- ★ Sparse computational kernels
- ★ Less memory requirements and possibly faster
- ★ Possible high weak scalability

Hybrid solver: Motivations

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

Usual trades off

Direct

- ★ Robust/prescribed accurate for general problems
- ★ BLAS-3 based implementations
- ★ Memory/CPU prohibitive for large $3D$ problems
- ★ Limited weak scalability

Iterative

- ★ Problem dependent efficiency / monitored accuracy
- ★ Sparse computational kernels
- ★ Less memory requirements and possibly faster
- ★ Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- Robust/prescribed accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems
- Limited weak scalability

Iterative

- Problem dependent efficiency / monitored accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster
- Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- ★ Robust/prescribed accurate for general problems
- ★ BLAS-3 based implementations
- ★ Memory/CPU prohibitive for large $3D$ problems
- ★ Limited weak scalability

Iterative

- ★ Problem dependent efficiency / monitored accuracy
- ★ Sparse computational kernels
- ★ Less memory requirements and possibly faster
- ★ Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

<u>Direct</u>

- Robust/prescribed accurate for general problems
- BLAS-3 based implementations
- Memory/CPU prohibitive for large $3D$ problems
- Limited weak scalability

<u>Iterative</u>

- Problem dependent efficiency / monitored accuracy
- Sparse computational kernels
- Less memory requirements and possibly faster
- Possible high weak scalability

Hybrid solver: Motivations

Goal: solving $\mathcal{A}x = b$, where $\mathcal{A}$ is large and sparse

Usual trades off

Direct

- ⋆ Robust/prescribed accurate for general problems
- ⋆ BLAS-3 based implementations
- ⋆ Memory/CPU prohibitive for large $3D$ problems
- ⋆ Limited weak scalability

Iterative

- ⋆ Problem dependent efficiency / monitored accuracy
- ⋆ Sparse computational kernels
- ⋆ Less memory requirements and possibly faster
- ⋆ Possible high weak scalability

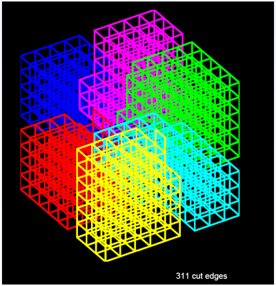
Governing Ideas in Hybrid Linear Solvers

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

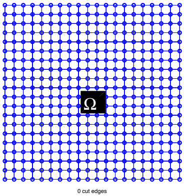
Domain Decomposition (DD)

- ⋆ 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

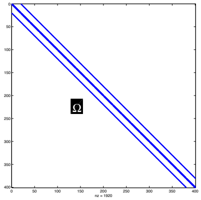


General partitioning of sparse matrix

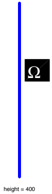
Mesh view



Matrix view



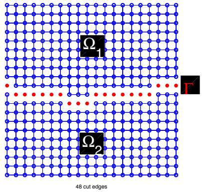
Tree view



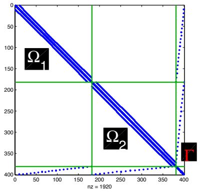
- ★ Partitioning a matrix using algebraic algorithm based on the adjacency graph of $\mathcal{A}$
- ★ 2 ways partitioning:
  - Computing an edge separator then finding the best vertex separator
  - Computing a vertex separator

General partitioning of sparse matrix

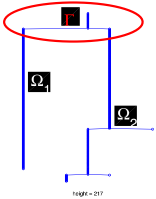
Mesh view



Matrix view



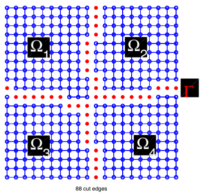
Tree view



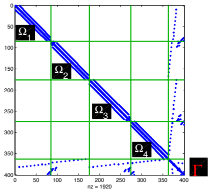
- ★ Partitioning a matrix using algebraic algorithm based on the adjacency graph of $\mathcal{A}$
- ★ 2 ways partitioning:
  - Computing an edge separator then finding the best vertex separator
  - Computing a vertex separator

General partitioning of sparse matrix

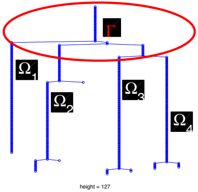
Mesh view



Matrix view



Tree view



MaPHyS

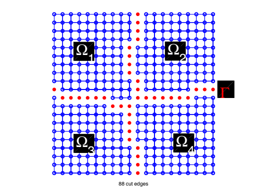
Global algebraic view

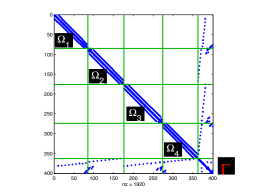
- ⋆ Global hybrid decomposition:

$$\mathcal{A} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}\mathcal{I}} & \mathcal{A}_{\mathcal{I}\Gamma} \\ \mathcal{A}_{\Gamma\mathcal{I}} & \mathcal{A}_{\Gamma\Gamma} \end{pmatrix}$$

- ⋆ Global Schur complement:

$$\mathcal{S} = \mathcal{A}_{\Gamma\Gamma} - \mathcal{A}_{\Gamma\mathcal{I}}\mathcal{A}_{\mathcal{I}\mathcal{I}}^{-1}\mathcal{A}_{\mathcal{I}\Gamma}$$





MaPHyS

Local algebraic view

- Local hybrid decomposition:

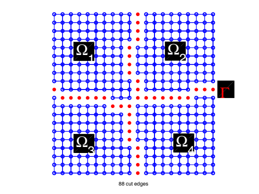
$$\mathcal{A}^{(i)} = \begin{pmatrix} \mathcal{A}_{\mathcal{I}_i\mathcal{I}_i} & \mathcal{A}_{\mathcal{I}_i\Gamma_i} \\ \mathcal{A}_{\Gamma_i\mathcal{I}_i} & \mathcal{A}^{(i)}_{\Gamma\Gamma} \end{pmatrix}$$

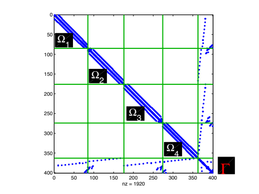
- Local Schur Complement:

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

- Algebraic Additive Schwarz Preconditioner

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





MaPHyS

Algebraic Additive Schwarz Preconditioner [L.Carvalho, L.Giraud, G.Meurant 01]

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

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

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

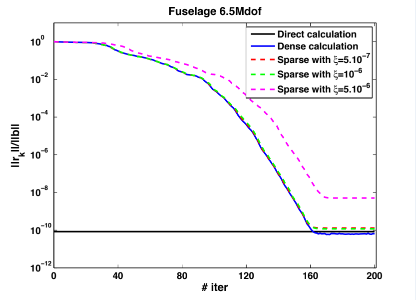
where $\bar{\mathcal{S}}^{(i)}$ is obtained from $\mathcal{S}^{(i)}$ via neighbor to neighbor comm

Closely related work

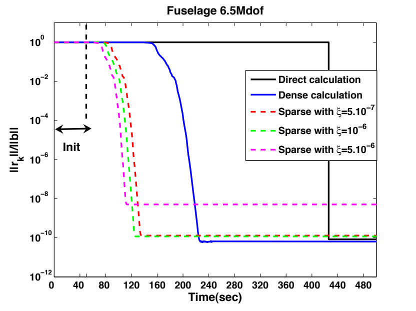
Similarity with Neumann-Neumann preconditioner [J.F Bourgat, R. Glowinski, P. Le Tallec and M. Vidrascu - 89] [Y.H. de Roek, P. Le Tallec and M. Vidrascu - 91]

Numerical behaviour of sparse preconditioners

Convergence history



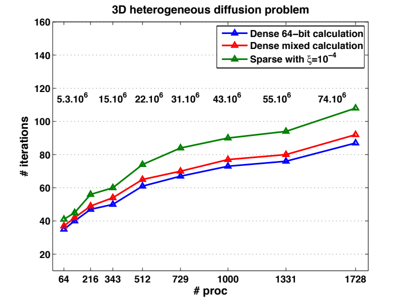
Time history



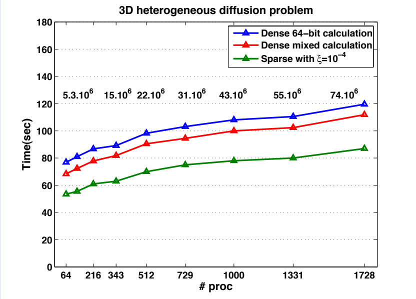
- ★ Fuselage problem of 6.5 Mdof dof mapped on 16 processors
- ★ The sparse preconditioner setup is 4 times faster than the dense one (19.5 v.s. 89 seconds)
- ★ In term of global computing time, the sparse algorithm is about twice faster
- ★ The attainable accuracy of the hybrid solver is comparable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

Scaled scalability on massively parallel platforms

Numerical scalability



Parallel performance

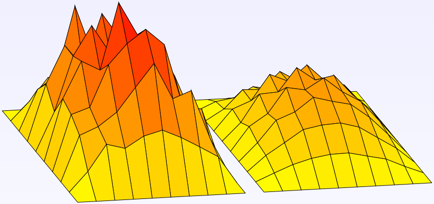


- ★ The solved problem size varies from 2.7 up to 74 Mdof
- ★ Control the growth in the # of iterations by introducing a coarse space correction
- ★ The computing time increases slightly when increasing # sub-domains
- ★ Although the preconditioners do not scale perfectly, the parallel time scalability is acceptable
- ★ The trend is similar for all variants of the preconditioners using CG Krylov solver

Outline

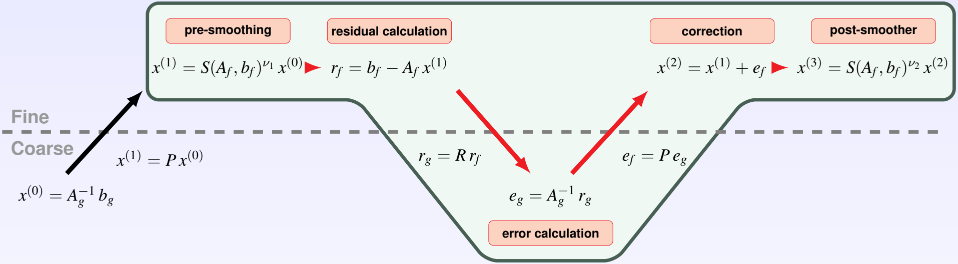
Multigrid Underlying ideas

- Stationary iteratve schemes versus the error modes
  - The high frequency modes are damped quickly
  - The low frequency modes are damped very slowly



- The low frequency modes might be viewed as high frequency on a coarser mesh

Full-Multigrid

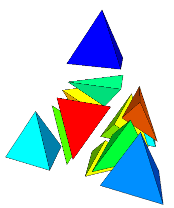


- Factorization of coarse problem performed only once
- Many forward/backward substitutions on coarsest grid
- Grid transfer operators to move within hierachical meshes
  - Fine $\rightarrow$ coarse : restriction
  - Coarse $\rightarrow$ fine : prolongation

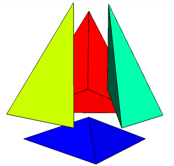
Refinement/Coarsening strategies

Unstructured mesh refinement

- ⋆ Isotropic refinment
  - \+ Preserve aspect ration
  - − Large number of fine elements (x12)
  - − Coupling interface change



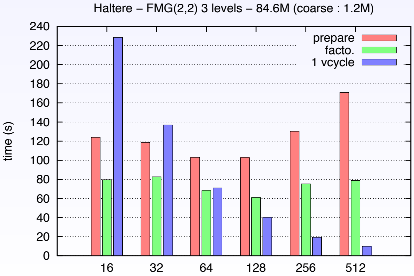
- ⋆ Anisotropic refinement
  - \+ Coupling interface unchanged
  - \+ Slower increase of mesh size x4
  - − Bad aspect ratio



Strong scalability

Table: fine 84.6M – coarse 1.2 M – FMG(2,2) 3 levels

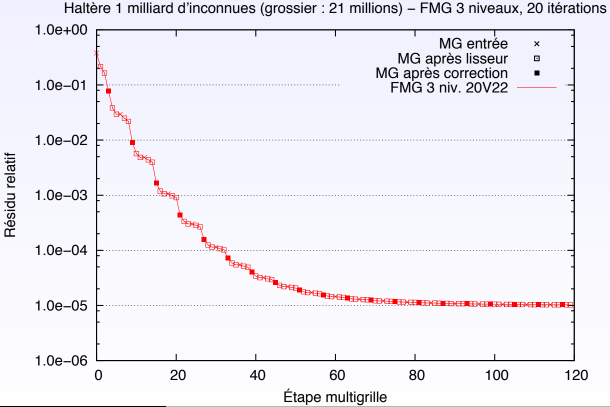
Cores	Init.	Assemb.	Facto.	Solve	10V
16	123.98	0.42	79.62	0.55	2285.58
32	118.67	0.23	82.72	0.30	1369.26
64	103.10	0.12	68.18	0.19	710.57
128	102.69	0.06	60.94	0.11	399.23
256	130.32	0.03	75.27	0.21	194.42
512	170.97	0.01	78.81	0.06	100.17



Validation – Strong scaling

Table: fine 1.3B – coarse 21.1 M – FMG(2,2) 3 levels

Cores	Init.	Assemb.	Facto.	Solve	10V
1024	1160.28	0.14	147.73	0.55	857.69



Outline

Concluding remarks

What I haven't talked about, but we could !!

- ⋆ Dense linear algebra kernels on emerging platforms, numerical resilient algorithms, iterative solvers for multiples right-hand sides, eigensolvers
- ⋆ Code coupling
- ⋆ Material physics

Sotware and collaborations

- ⋆ Available packages : HIPS, MaPHys, ScalFMM, EPSN
- ⋆ Ongoing collaborations: MORSE (KAUST, UCD, UTK), FAST-LA (LBNL, Stanford)

Concluding remarks

What I haven't talked about, but we could !!

- Dense linear algebra kernels on emerging platforms, numerical resilient algorithms, iterative solvers for multiples right-hand sides, eigensolvers
- Code coupling
- Material physics

Sotware and collaborations

- Available packages : HIPS, MaPHys, ScalFMM, EPSN
- Ongoing collaborations: MORSE (KAUST, UCD, UTK), FAST-LA (LBNL, Stanford)

Advert

Post-doc position

One year post-doc position on resilient numerical algorithms in the framework of the G8-ECS project (Enabling Climate Simulations at Extreme Scale), to be fulfiled asap.

Advert

Post-doc position

One year post-doc position on resilient numerical algorithms in the framework of the G8-ECS project (Enabling Climate Simulations at Extreme Scale), to be fulfiled asap.

Obrigado for your attention

Advert

Post-doc position

One year post-doc position on resilient numerical algorithms in the framework of the G8-ECS project (Enabling Climate Simulations at Extreme Scale), to be fulfiled asap.

Obrigado for your attention

Questions ?