Modélisation de la propagation des ondes sismiques en multi-GPUs

Dimitri Komatitsch, David Michéa and Roland Martin
(Univ Pau, CNRS and INRIA Sud-Ouest Magique3D)

Gordon Erlebacher (Department of Scientific Computing, Florida State University, USA)

Dominik Göddeke (TU Dortmund, Germany)

GDR Calcul, Lyon
November 9, 2010

6 April 2009
M_w 6.2 L'Aquila (Italy)

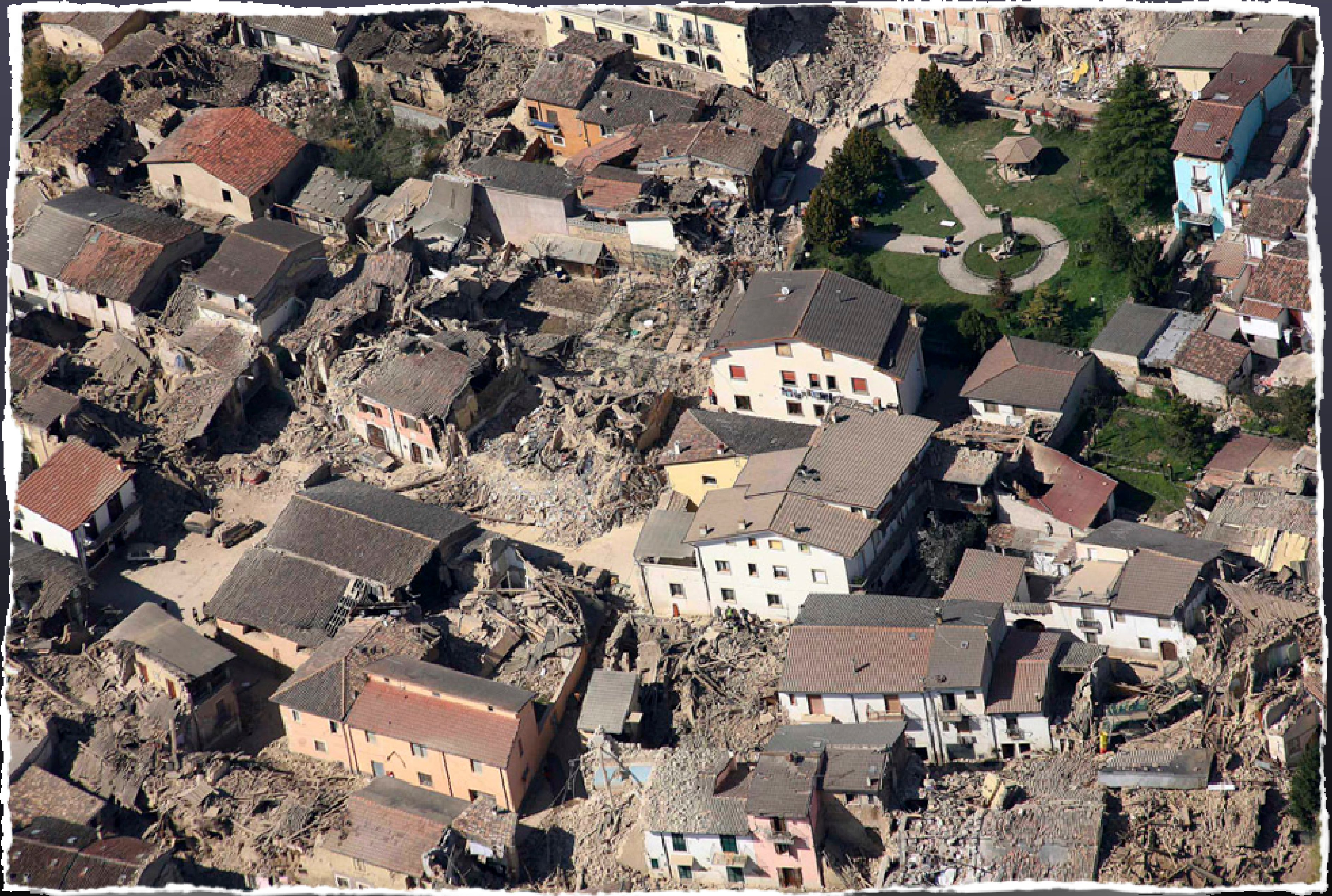
260 dead.
~1.000 hurts
~ 26.000 homeless

Istituto Nazionale di Geofisica e Vulcanologia

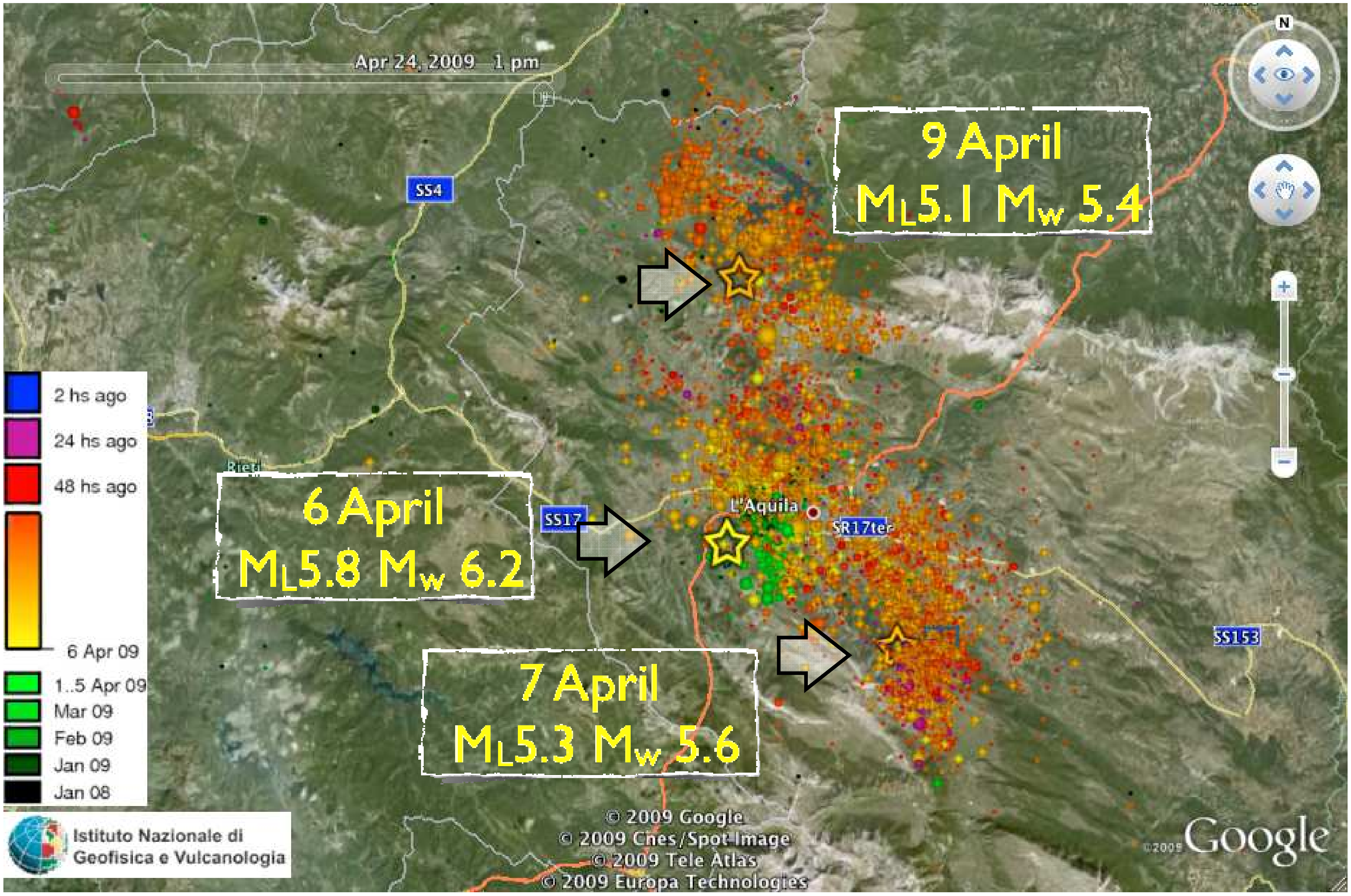
Collaboration with Emanuele Casarotti (INGV, Roma, Italy)

M_w 6.2 L'Aquila

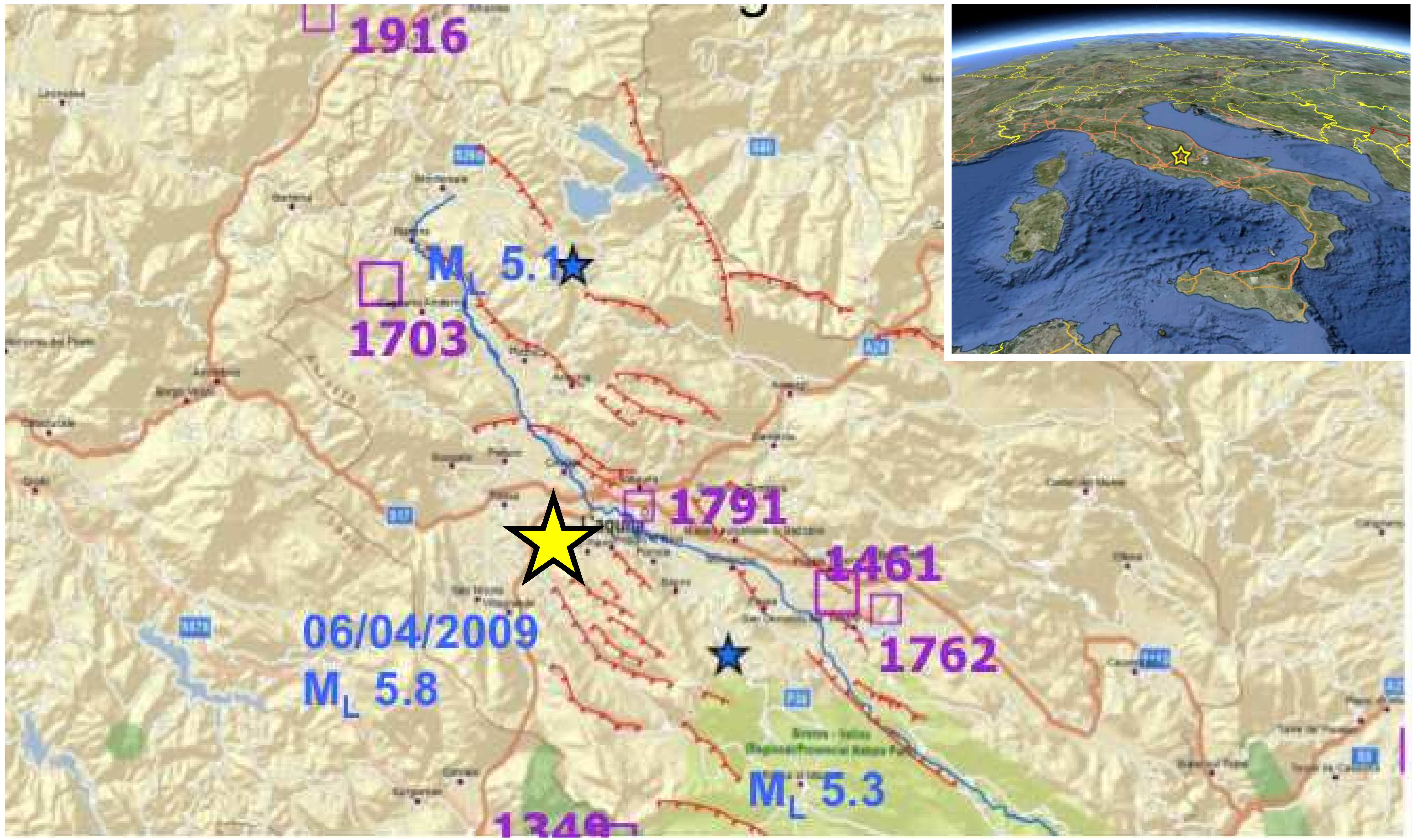
M_w 6.2 L'Aquila



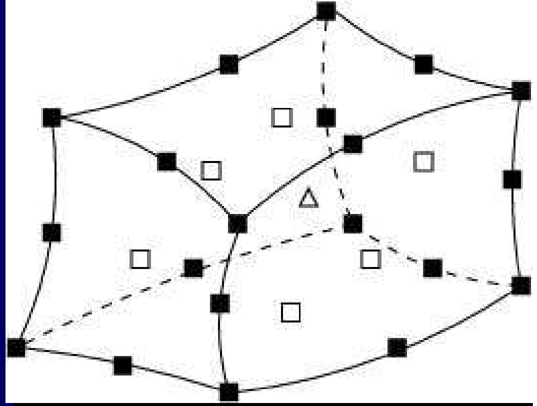
"Mainshocks and Aftershocks"



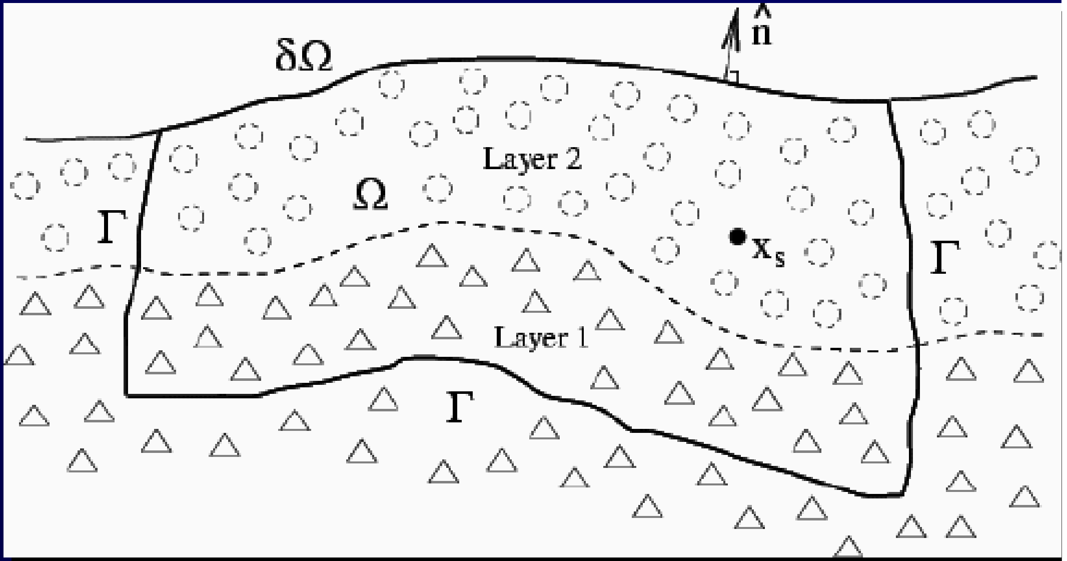
Where

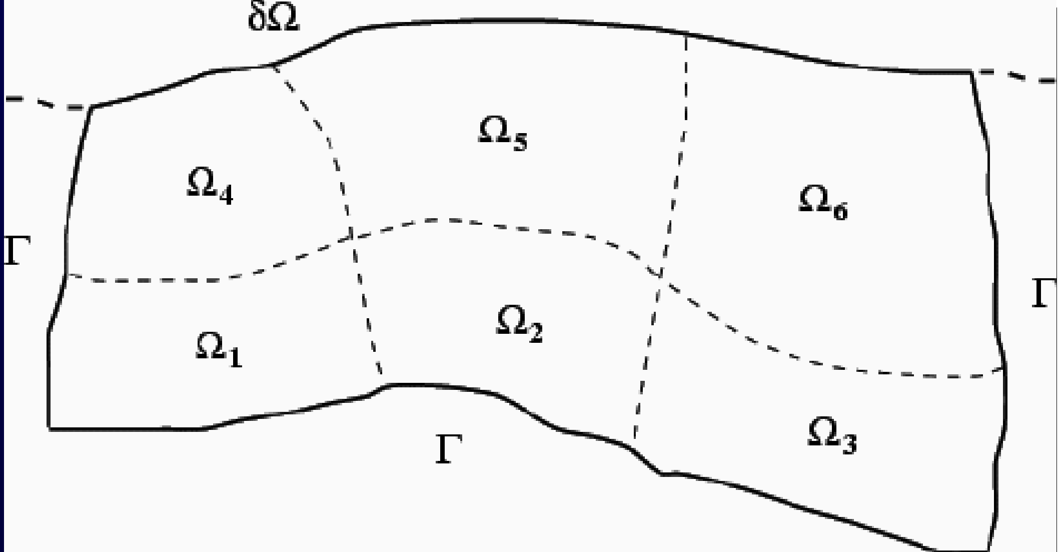


Spectral-Element Method



- Developed in Computational Fluid Dynamics (Patera 1984, Maday and Patera 1988, 1989…)
- Accuracy of a pseudospectral method, flexibility of a finite-element method
- Extended by Komatitsch and Tromp, Capdeville et al.
- Large curved "spectral" finite-elements with high-degree polynomial interpolation
- Mesh honors the main discontinuities (velocity, density) and topography
- Very efficient on parallel computers, no linear system to invert (diagonal mass matrix)
- No need for Discontinuous Galerkin





Equations of motion (solid)

Differential or *strong* form (e.g., finite differences):

$$\rho \partial_t^2 \mathbf{s} = \nabla \cdot \sigma + \mathbf{f}$$

We solve the integral or *weak* form:

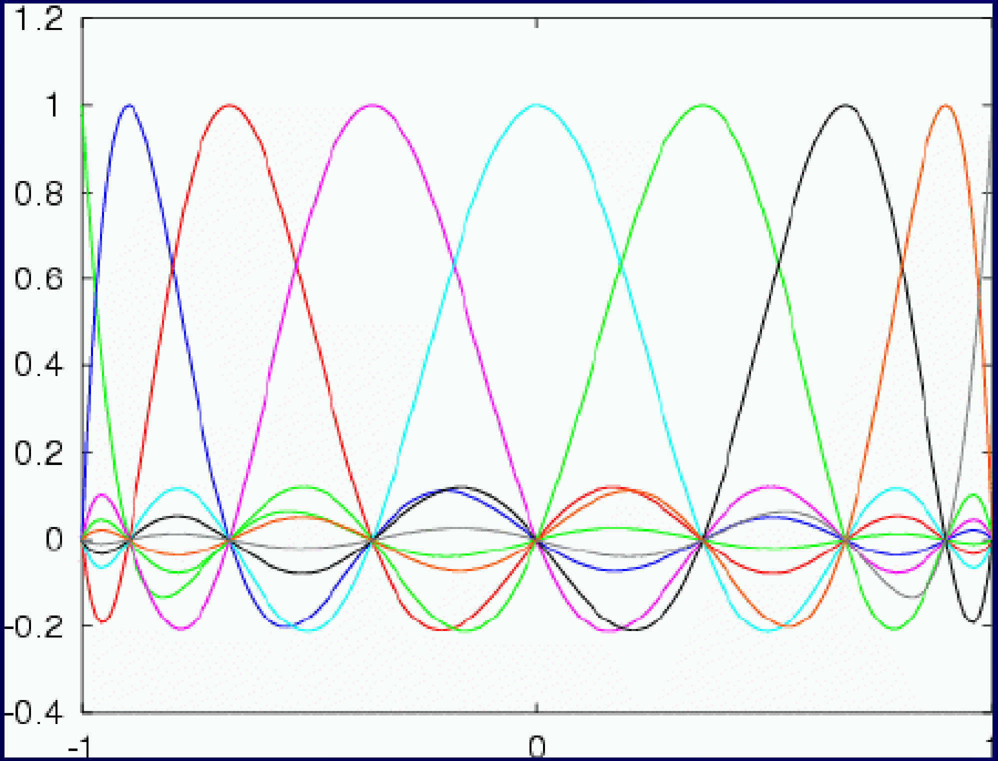
$$\int \rho \, \mathbf{w} \cdot \partial_t^2 \mathbf{s} \, \mathrm{d}^3\mathbf{r} = -\int \nabla \mathbf{w} : \sigma \, \mathrm{d}^3\mathbf{r}$$

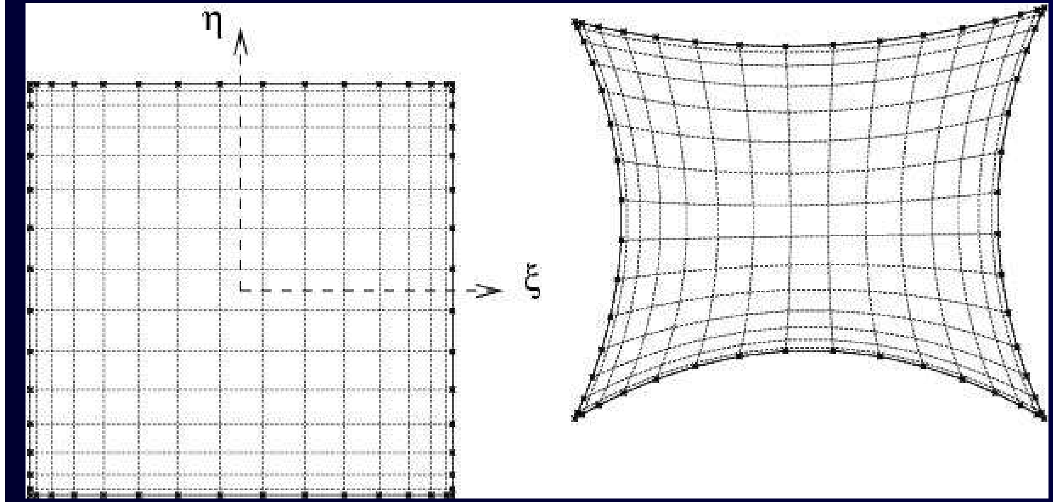
$$+ \mathbf{M} : \nabla \mathbf{w}(\mathbf{r}_\mathrm{s}) S(t) - \int_{\mathrm{F-S}} \mathbf{w} \cdot \sigma \cdot \hat{\mathbf{n}} \, \mathrm{d}^2\mathbf{r}$$

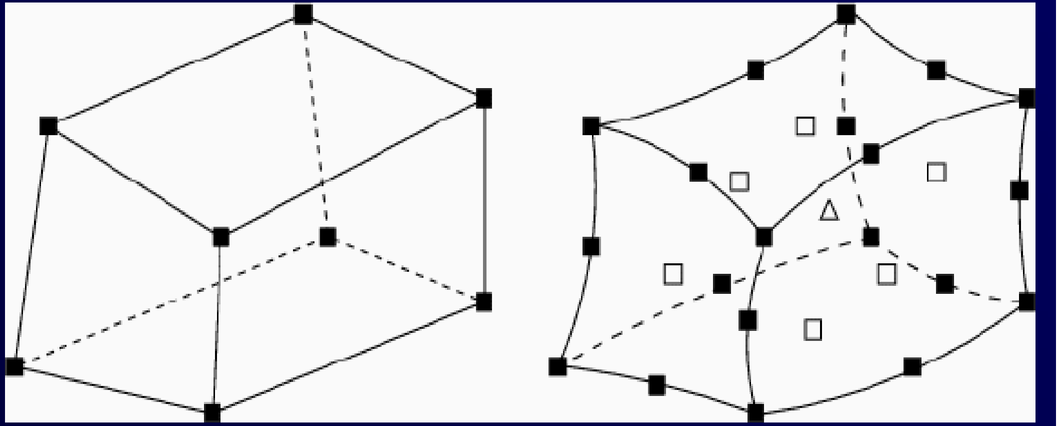
+ attenuation (memory variables) and ocean load

Finite elements

- High-degree pseudospectral finite elements
- N = 5 to 8 usually
- *Exactly* Diagonal mass matrix
- No linear system to invert







Global Simulations: SPECFEM3D_GLOBE

Open source: geodynamics.org

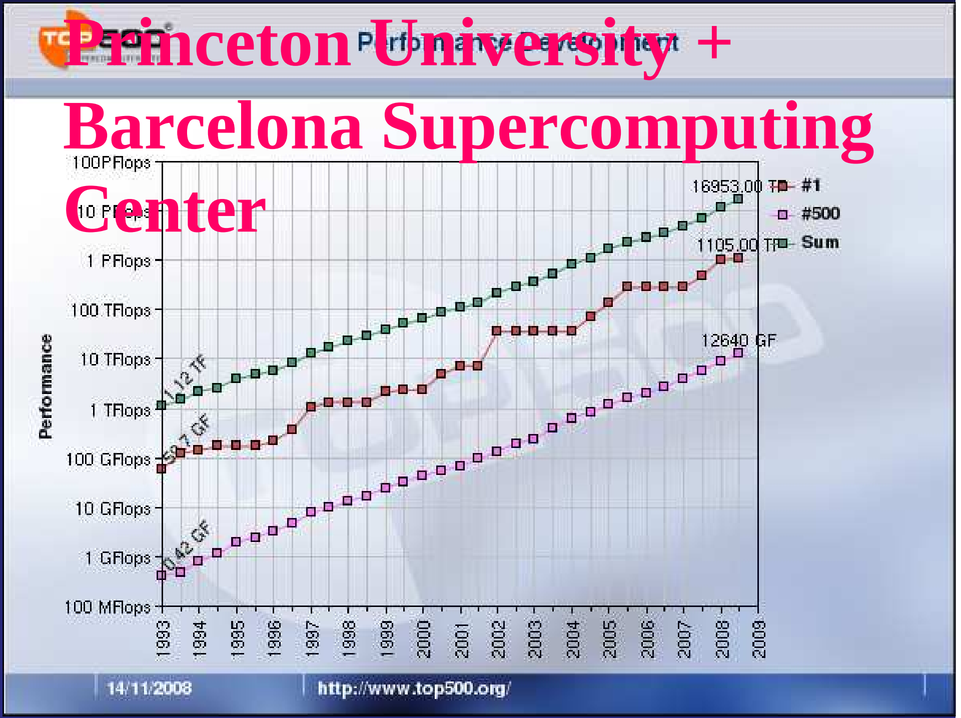
On-demand TeraGrid applications:

- Automated, near real-time simulations of all M>6 earthquakes
- Analysis of past events (more than 20,000 events)
- Seismology Web Portal (geodynamics.org)

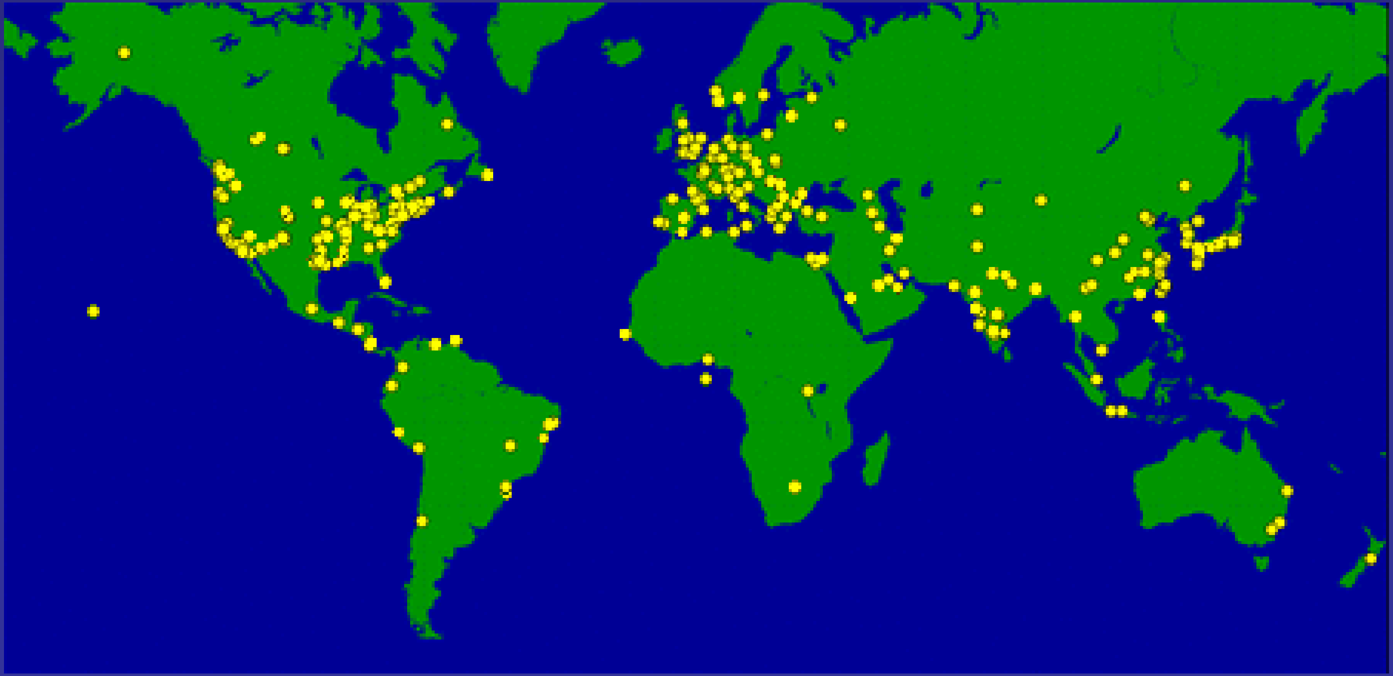
Petascale simulations:

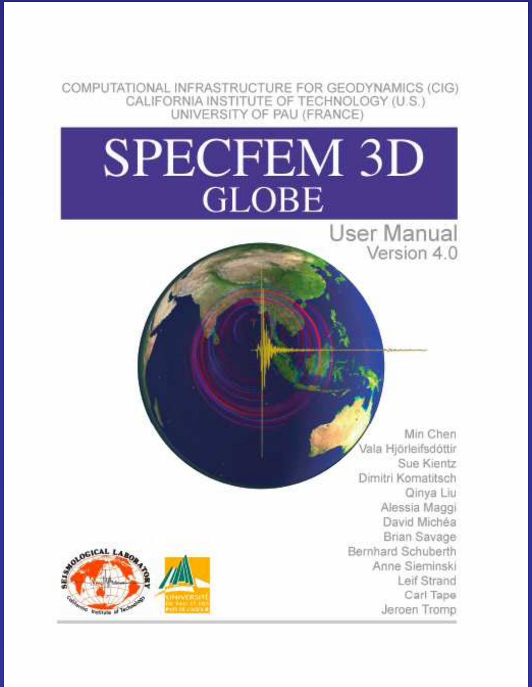
- Global simulations at 1-2 Hz
- Reached 1.15 s period on 149,784 cores at ORNL
- Moving towards global 'adjoint tomography'

Princeton University + Barcelona Supercomputing Center



SPECFEM3D_GLOBE Users Map





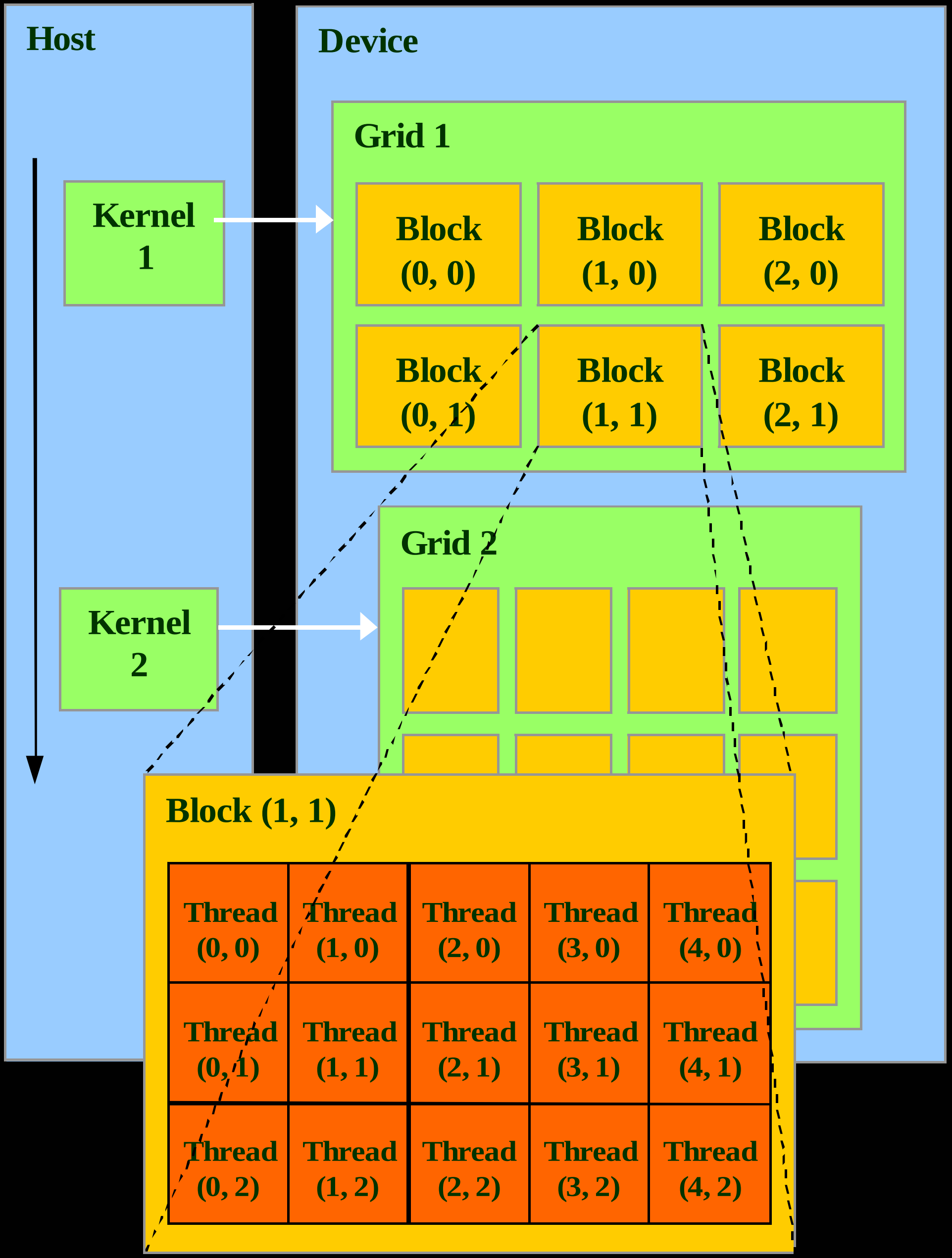
COMPUTATIONAL INFRASTRUCTURE FOR GEODYNAMICS (CIG)
CALIFORNIA INSTITUTE OF TECHNOLOGY (U.S.)
UNIVERSITY OF PAU (FRANCE)

SPECFEM 3D
GLOBE

User Manual
Version 4.0

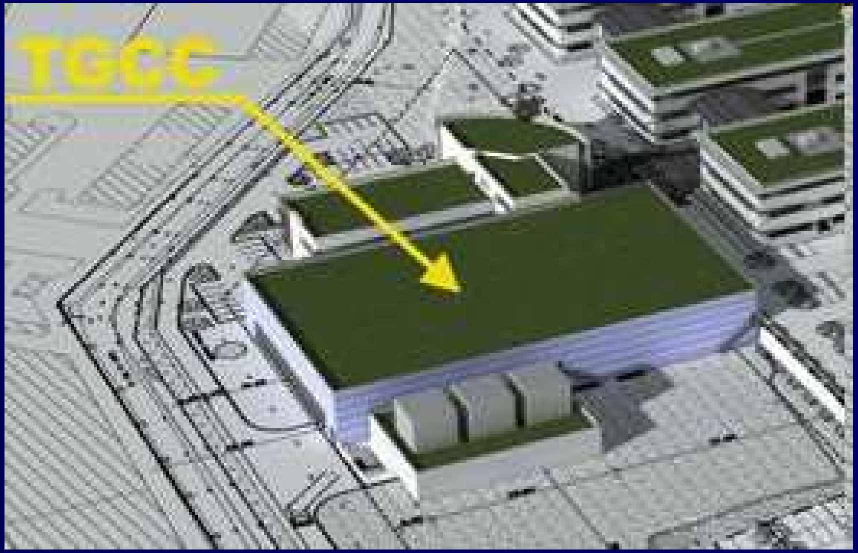
Min Chen
Vala Hjörleifsdóttir
Sue Kientz
Dimitri Komatitsch
Qinya Liu
Alessia Maggi
David Michéa
Brian Savage
Bernhard Schuberth
Anne Sieminski
Leif Strand
Carl Tape
Jeroen Tromp

Graphics cards NVIDIA GeForce 8800 GTX



Why are they so powerful for scientific computing?

TGCC + PRACE



- Salles informatiques : 2600 m2
- Alimentation électrique : ligne de 60 MW
- L'échelle du petaflop pour le calculateur de la future infrastructure Européenne

http://www.teratec.eu/technopole/tgcc.html

Le TGCC (Très Grand Centre de Calcul) sera disponible en 2010 pour accueillir la machine Européenne PRACE financée par GENCI.

GENCI (Grand Équipement National de Calcul Intensif)

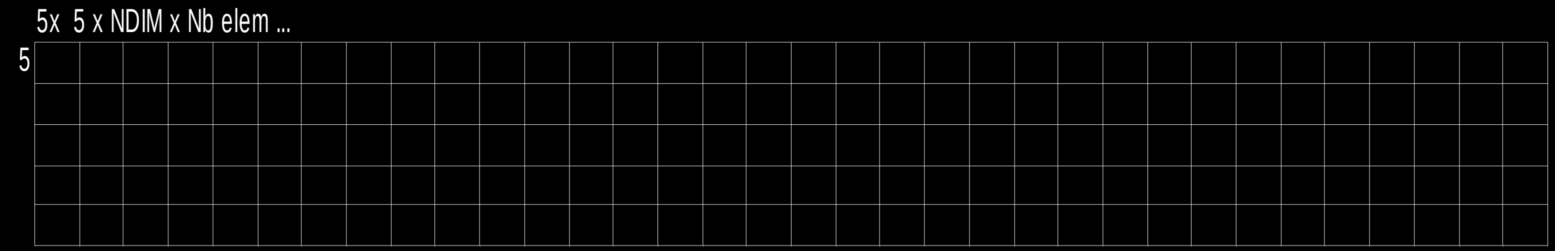
Machine chinoise n°1 au monde

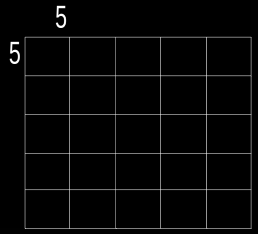
http://online.wsj.com

- Tianhe-1A supercomputer
- located at National Supercomputer Center in Tianjin, China
- GPU based 2.5 petaflop supercomputer
- Will likely be number 1 in the next Top500 list of the fastest supercomputers

Notons également que le matériel s'améliore très rapidement : NVIDIA Fermi, Maxwell, ATI/AMD, et les logiciels de support également : OpenCL au lieu de CUDA.

BLAS 3 (Basic Linear Algebra Subroutines)





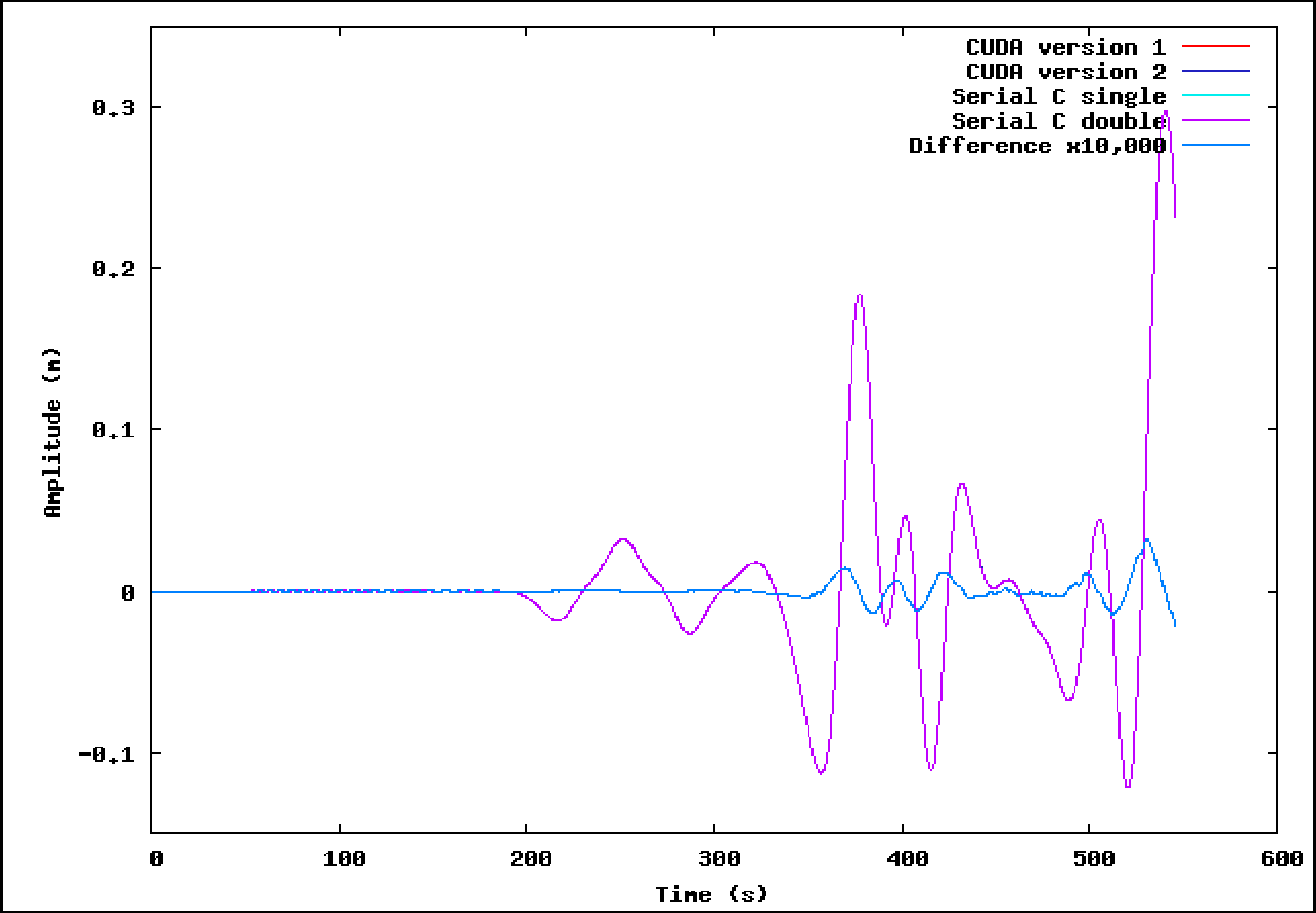
Can we use highly optimized BLAS matrix/matrix products (90% of computations)?

- For one element: matrices (5x25, 25x5, 5 x matrices of (5x5)), BLAS is not efficient: overhead is too expensive for matrices smaller than 20 to 30 square.
- If we build big matrices by appending several elements, we have to build 3 matrices, each having a main direction (x,y,z), which causes a lot of cache misses due to the global access because the elements are taken in different orders, thus destroying spatial locality.
- Since all arrays are static, the compiler already produces a very well optimized code.

=> No need to, and cannot easily use BLAS

=> Compiler already does an excellent job for small static loops

Porting SPECFEM3D on CUDA: validation



Minimize CPU ↔ GPU data transfers

- CPU ↔ GPU memory bandwidth much lower than GPU memory bandwidth
  - Use page-locked host memory (`cudaMallocHost()`) for maximum CPU ↔ GPU bandwidth

- Minimize CPU ↔ GPU data transfers by moving more code from CPU to GPU
  - Even if that means running kernels with low parallelism computations
  - Intermediate data structures can be allocated, operated on, and deallocated without ever copying them to CPU memory

- Group data transfers
  - One large transfer much better than many small ones

- Fit all the arrays on the GPU card to avoid costly CPU ↔ GPU data transfers

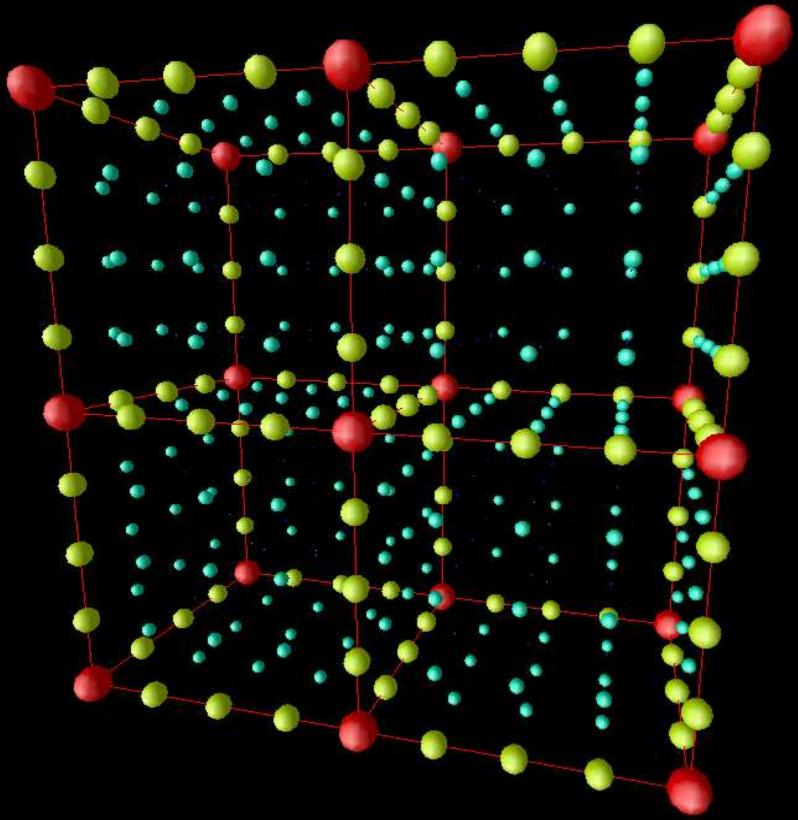
- But of course the MPI buffers must remain on the CPU, therefore we can not avoid a small number of transfers (of 2D cut planes)

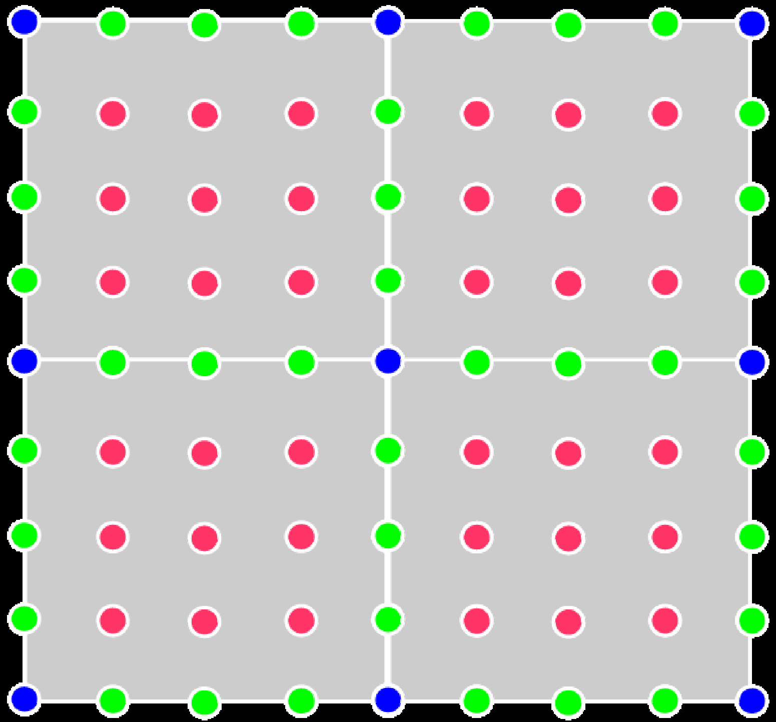
Porting SPECFEM3D on CUDA

- At each iteration of the serial time loop, three main types of operations are performed:
  - **update (with no dependency) of some global arrays composed of the unique points of the mesh**
  - **purely local calculations of the product of predefined derivative matrices with a local copy of the displacement vector along cut planes in the three directions (i, j and k) of a 3D spectral element**
  - **update (with no dependency) of other global arrays composed of the unique points of the mesh**

Porting SPECFEM3D on CUDA: global numbering versus local numbering

- **In 3D and for NGLL = 5, for a regular hexahedral mesh there are:**
  - **125 GLL integration points in each element**
  - **27 belong only to this element**
  - **98 belong to several elements**



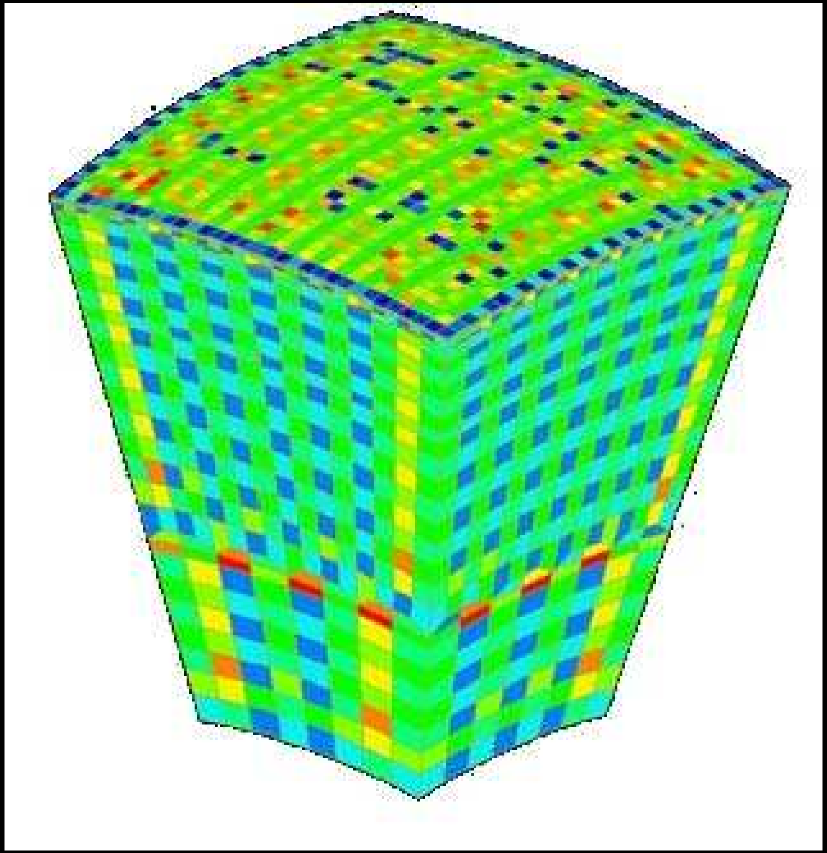


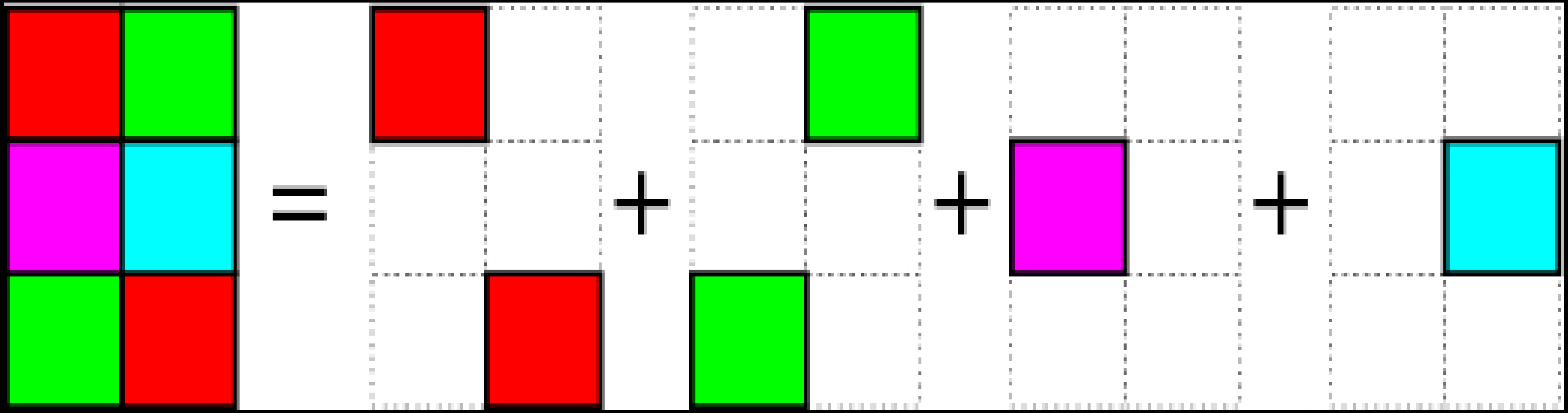
=> one thread per grid point (i.e., 125 threads per finite element)

Porting SPECFEM3D on CUDA: mesh coloring

- Key challenge: ensure that contributions from two local nodes never update the same global value from different warps

- Use of mesh coloring: suppress dependencies between mesh points inside a given kernel





Porting SPECFEM3D on CUDA: Coalesced Global Memory Accesses

- To ensure coalesced reads from global memory, the local array sizes are a multiple of 128 floats (which is itself a multiple of the half-warp size of 16) instead of 5^3 = 125 (thus purposely wasting 128/125 = 1.023 = 2.3% of memory)

- Each thread is responsible for a different point in the element. Consequently, the threads of a half-warp load adjacent elements of a (float) array. Access to global memory is thus perfectly coalesced in kernels 1 and 3, as well as in the parts of kernel 2 that access local arrays

- When accessing global arrays in kernel 2, the indirect addressing necessary to handle the unstructured mesh topology results in non-coalescent accesses and 5-way bank conflicts

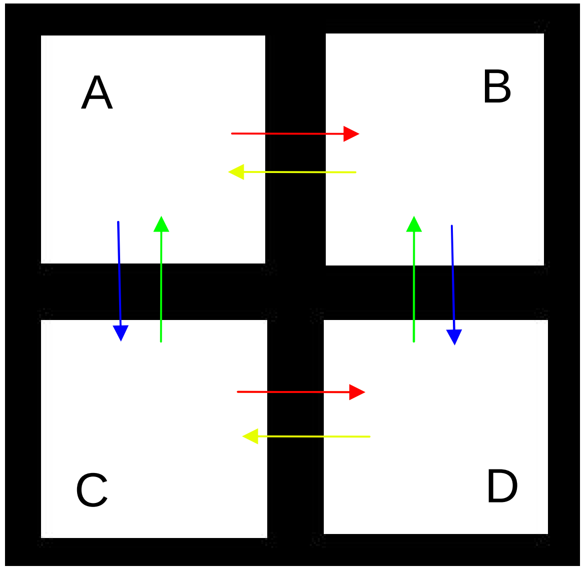
Porting SPECFEM3D on CUDA: Coalesced Global Memory Accesses

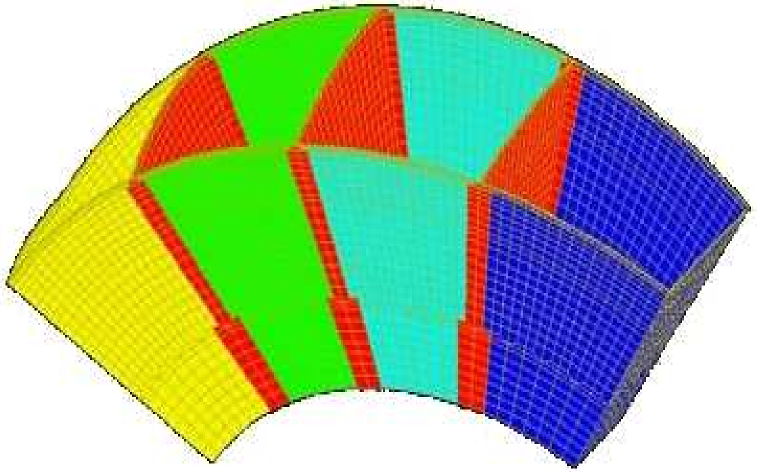
- In kernel 1 & 3, all accesses are perfectly coalesced
- In kernel 2, all accesses from local arrays are perfectly coalesced.
- When accessing global arrays in kernel 2, the indirect addressing necessary to handle the unstructured mesh topology results in non-coalescent accesses.
- This has become far less critical on FERMI

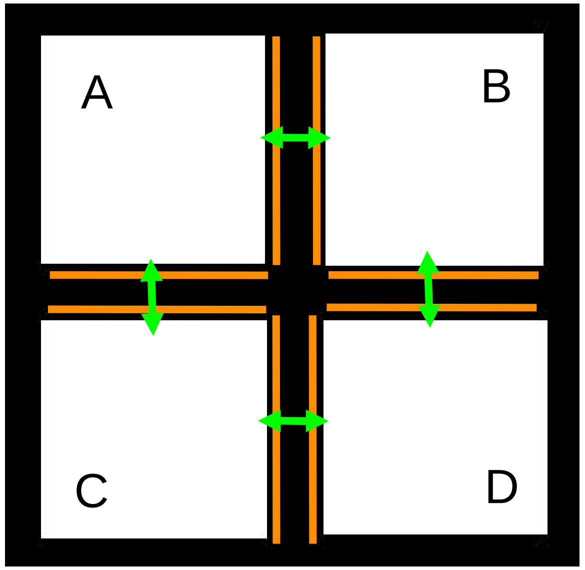
Porting SPECFEM3D on CUDA: adding MPI

Old communication scheme (blocking MPI)
Update done in the whole arrays
(all elements computed before starting MPI calls)

New communication scheme (non blocking MPI)
Update done in buffers (for outer mesh elements first)





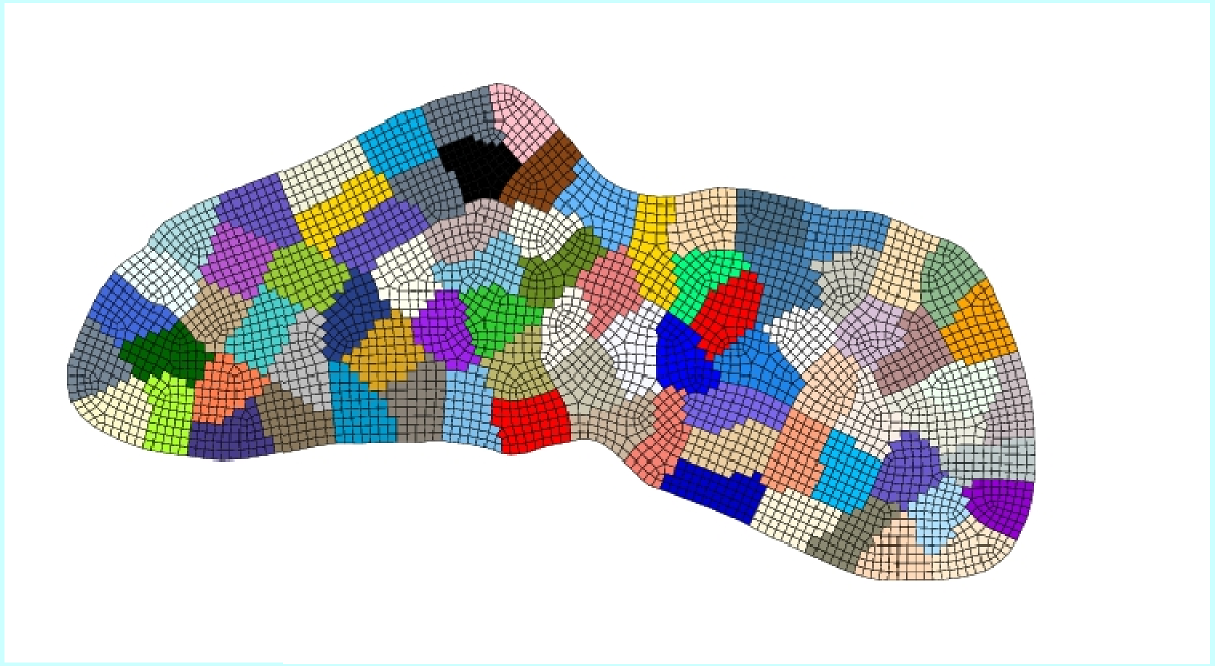


MPI communications cost on GPU version ~ 5%,

> We need to use non-blocking MPI communications.

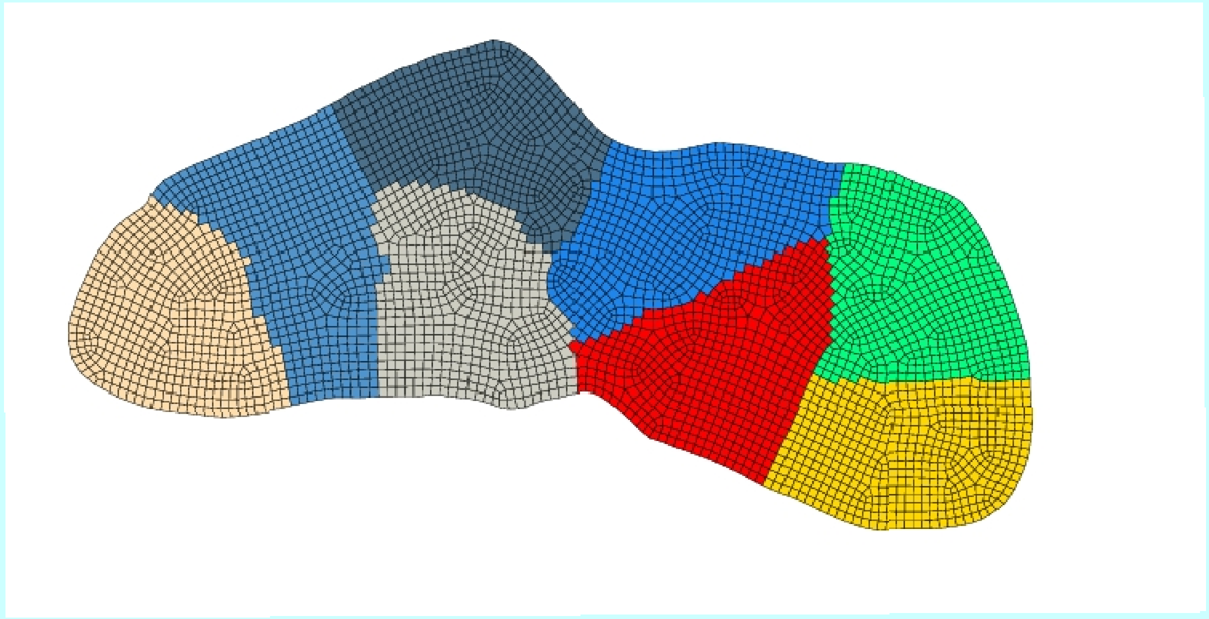
> MPI communications are very well overlapped by computations on the GPU.

Use non-blocking MPI



80 domaines : nombre équivalent d'éléments internes et aux interfaces

Danielson and Namburu (1998)



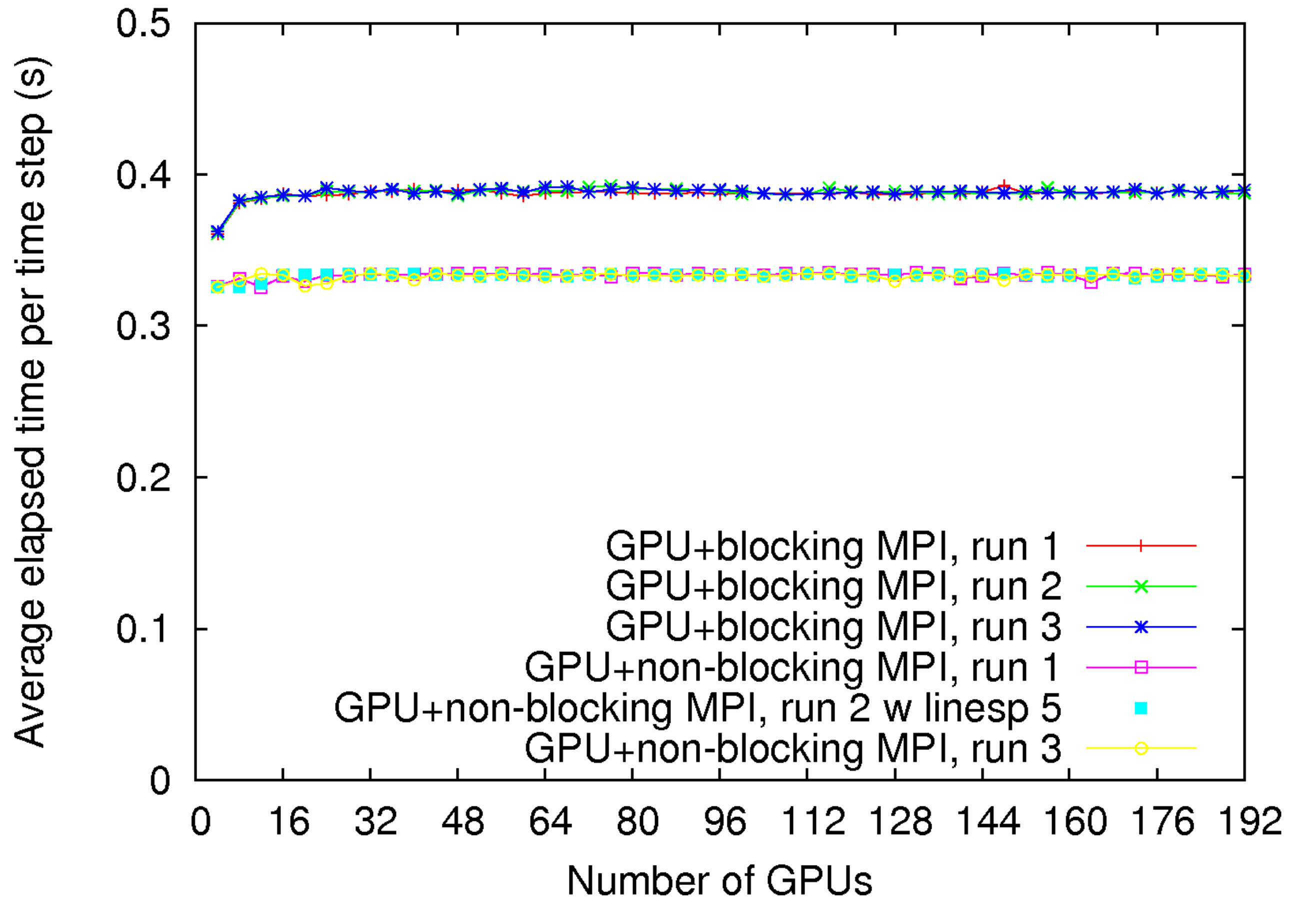
8 domaines : nombre d'éléments aux interfaces

< nombre d'éléments internes

Collaboration with Roland Martin and Nicolas Le Goff (Univ of Pau, France)

MultiGPU weak scaling (up to 192 GPUs)

technische universität dortmund

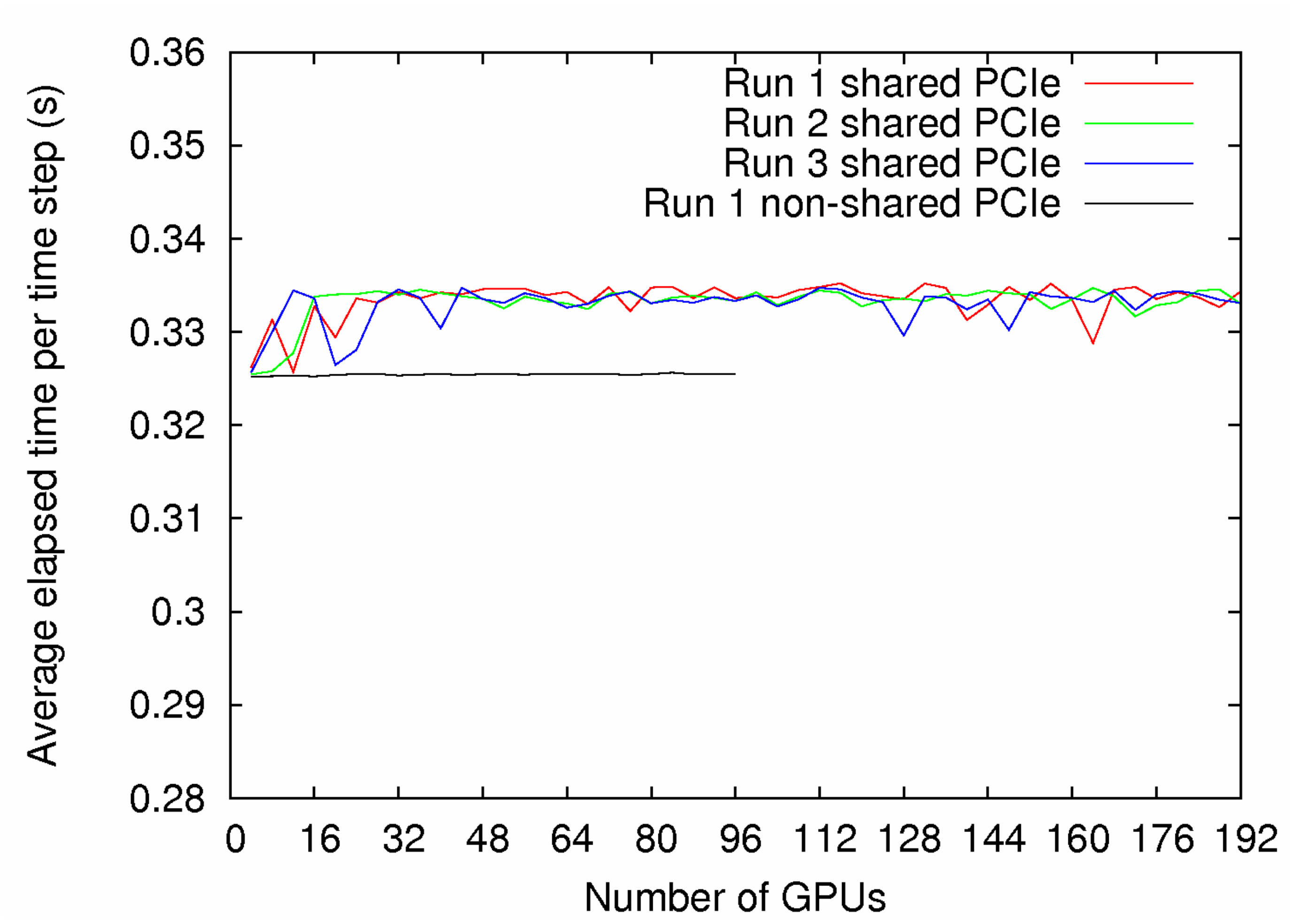


- Constant problem size of 3.6 GB per GPU
- Weak scaling excellent up to 17 billion unknowns
- Blocking MPI results in 20% slowdown

- It is difficult to define speedup: versus what?
- For us, on the CEA/CCRT/GENCI GPU/Nehalem cluster, **about 12x versus all the CPU cores, 20x for one GPU versus one CPU core.**

Effect of bus sharing

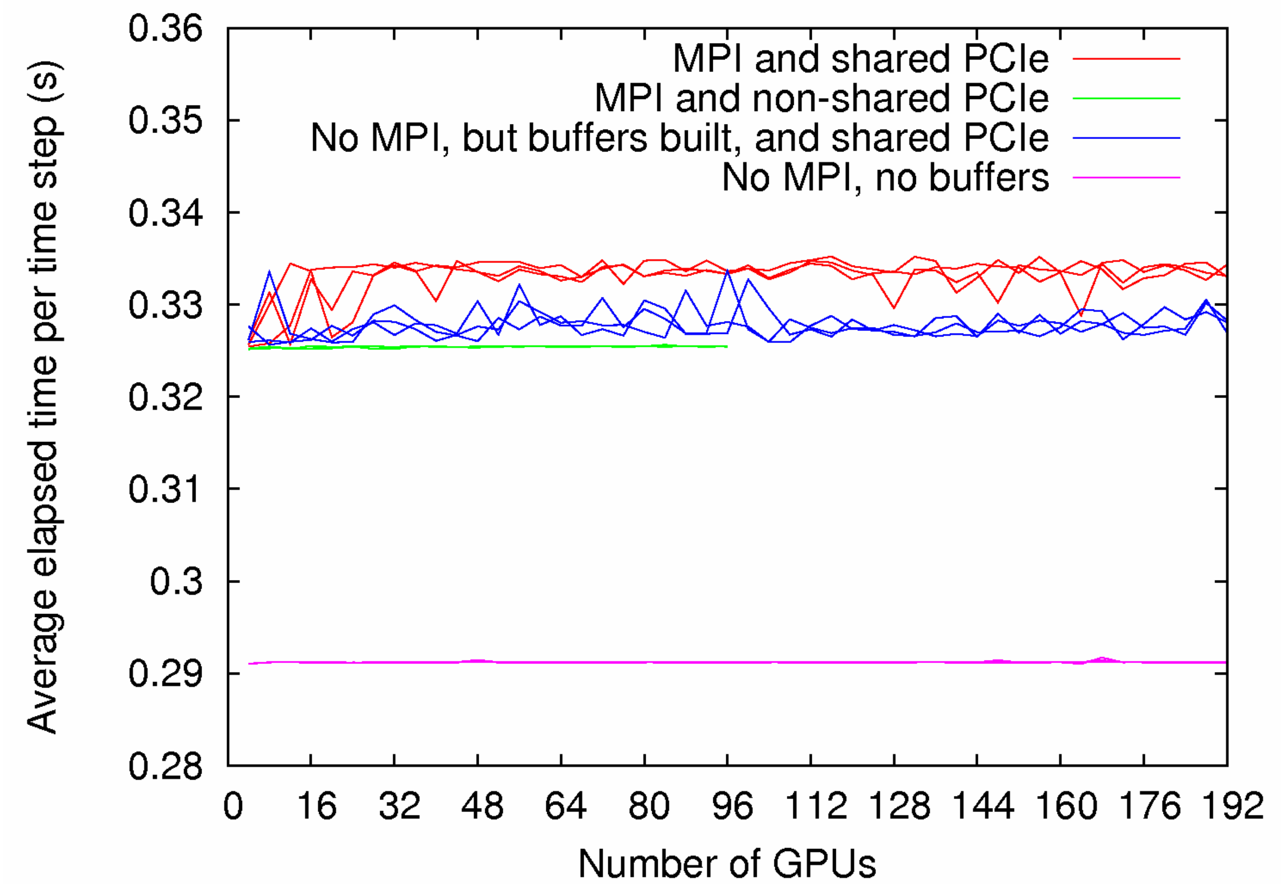
technische universität dortmund



- 2 GPUs share one PCIe bus in the Tesla S1070 architecture
- This is a potentially huge bottleneck!
- Bus sharing introduces fluctuations between runs and a slowdown ≤ 3%

GPU performance breakdown

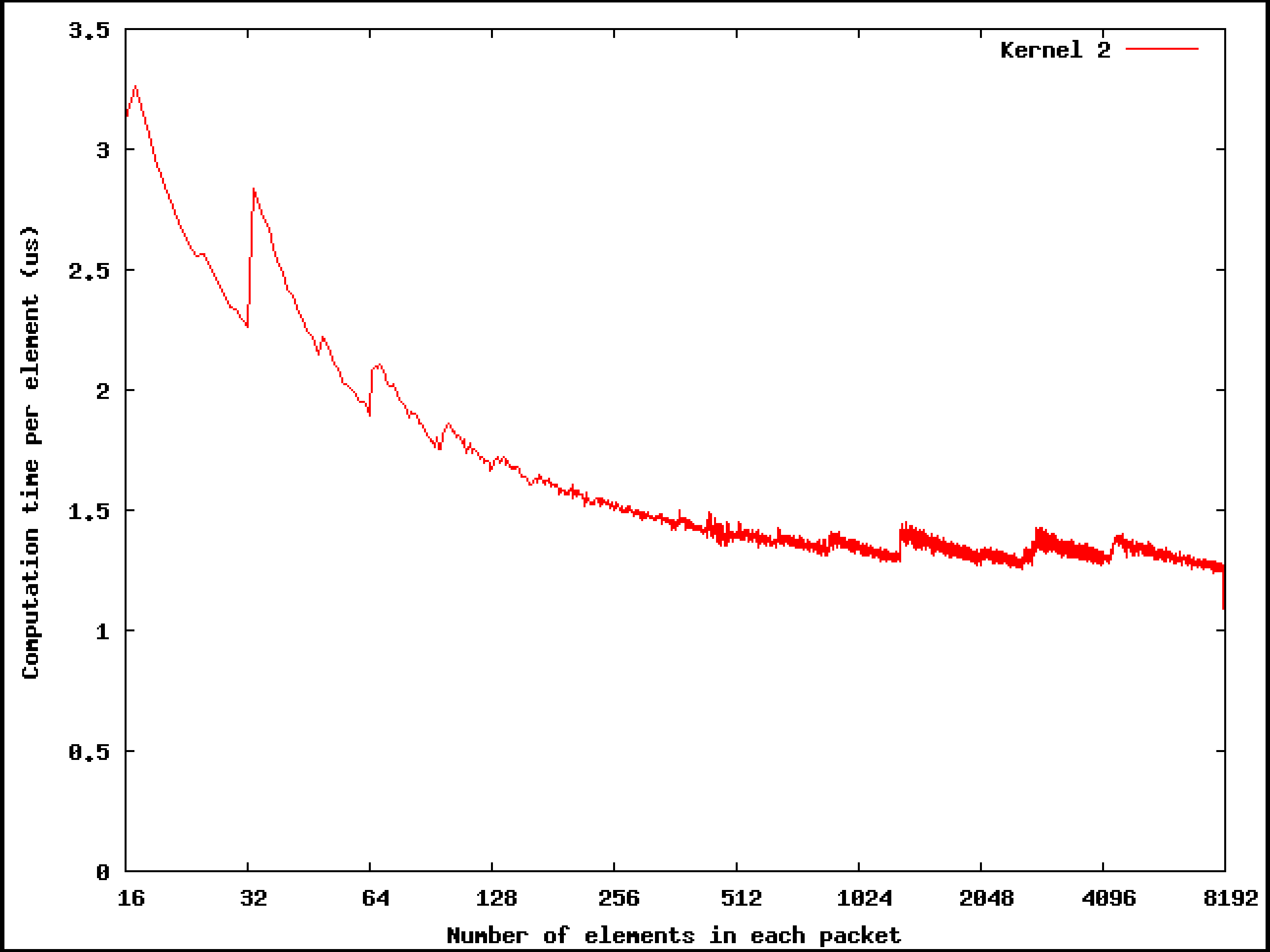
technische universität dortmund



- Effect of overlapping (no MPI = replace send/receive with zeroing)
- Red vs. blue curve: Difference ≤ 2.8%, i.e., very good overlap
- Green vs. magenta: Total overhead cost of running this problem on a cluster is ≤ 12% (for building, processing and transmitting buffers)

Porting SPECFEM3D on CUDA: results

Mesh size	GTX 280		8800 GTX				
	Version 1		Version 1		Version 2		
	Time / element	Speedup	Time / element	Speedup	Time / element	Speedup	Transfert ime
65 MB	0.94 µs	21.5	1.5 µs	13.5	4.2 µs	4.6	68%
405 MB	0.79 µs	24.8	1.3 µs	15	3.7 µs	5.3	68%
633 MB	0.77 µs	25.3	1.3 µs	15	3.7 µs	5.3	67%



- The final speedup for the CUDA + MPI code is 25x
- Performance evolution depends on parameters that are difficult to choose

Efforts pour encapsuler le code :

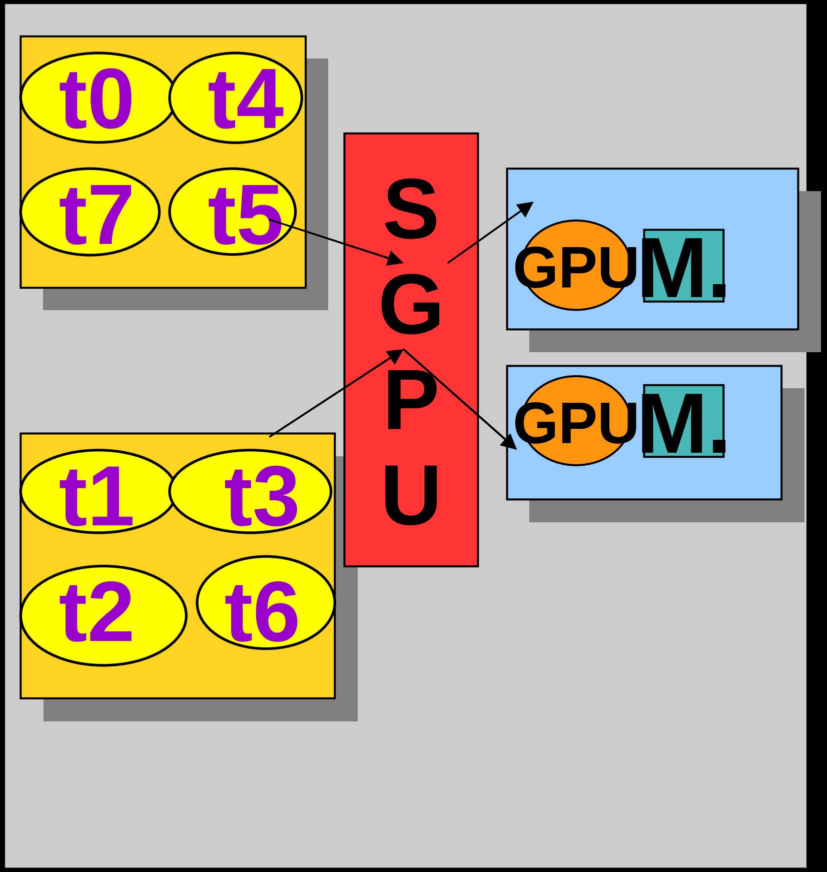
- Ils sont actuellement élevés : réécriture du code en CUDA de NVIDIA, OpenCL ou similaire.
- Exemple : SPECFEM3D, 12 à 18 mois à temps complet pour un ingénieur de recherche INRIA pour obtenir une première version, un an de plus pour l’intégrer à notre code de production

Aide possible de la communauté :

- Communauté nombreuse et active : France : GENCI, CEA, INRIA, CNRS, Académie, Groupe Calcul et GDR, ORAP, TOTAL, CERFACS etc…
- Efforts d’automatisation ou de standardisation dans la communauté : OpenCL, HMPP CAPS, StarSs (Barcelone), StarPU (INRIA Bordeaux), S_GPU (INRIA Grenoble), …

The S_GPU library

- Implemented by INRIA Grenoble
- Virtualization : 1 GPU visible per CPU core
- Instructions scheduling done by S_GPU, not by CUDA
- Memory transfers / computations overlapping
- Written in C++ and CUDA, binding Fortran
- Limited intrusion in the source code

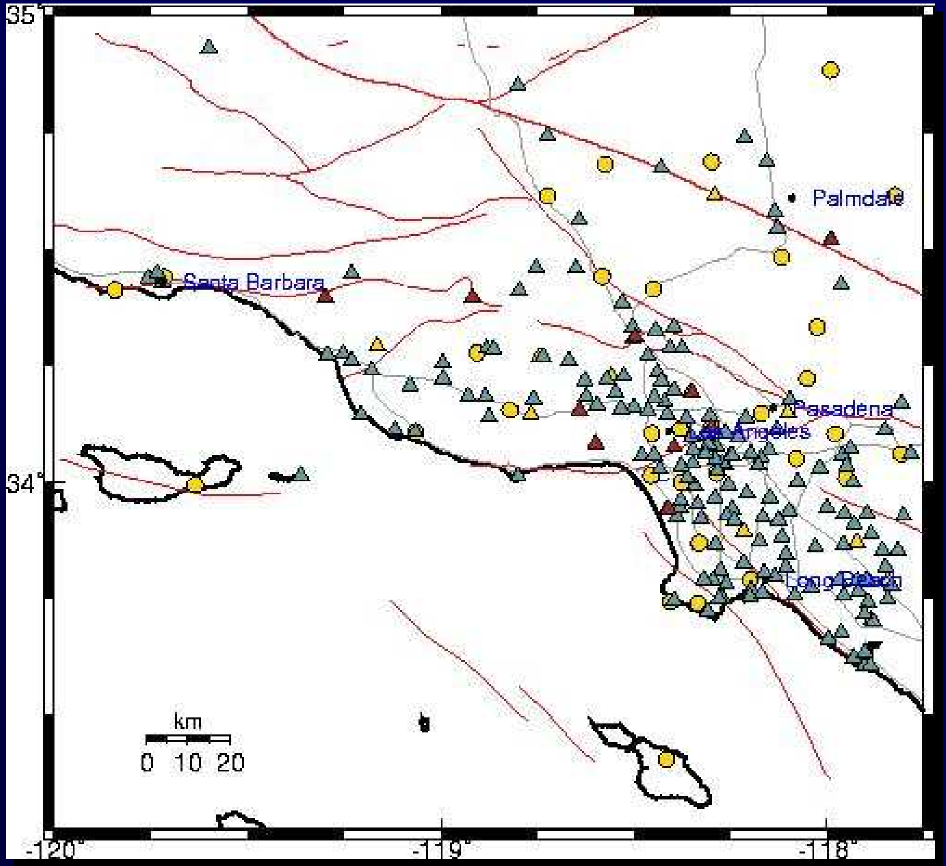


HMPP directives (INRIA CAPS)

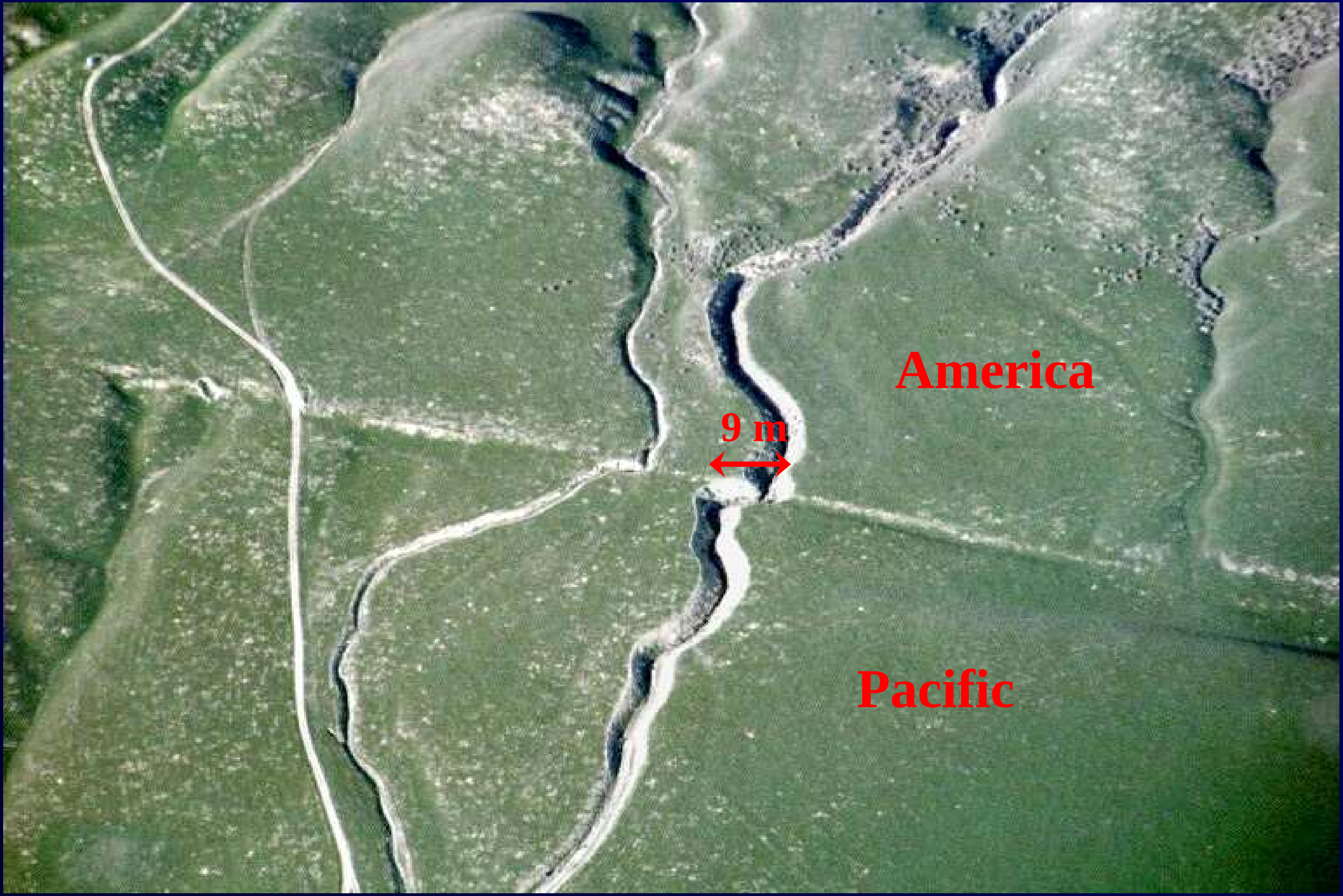
- Flexible model consisting of directives that express parallelism in the code and generate GPU or multicore code automatically

Wave propagation in basins

- Need accurate numerical methods to model seismic hazard – very densely populated areas
- Large and complex 3D models (e.g., L.A., Tokyo, Mexico)
- Wealth of high-quality data (TriNet)

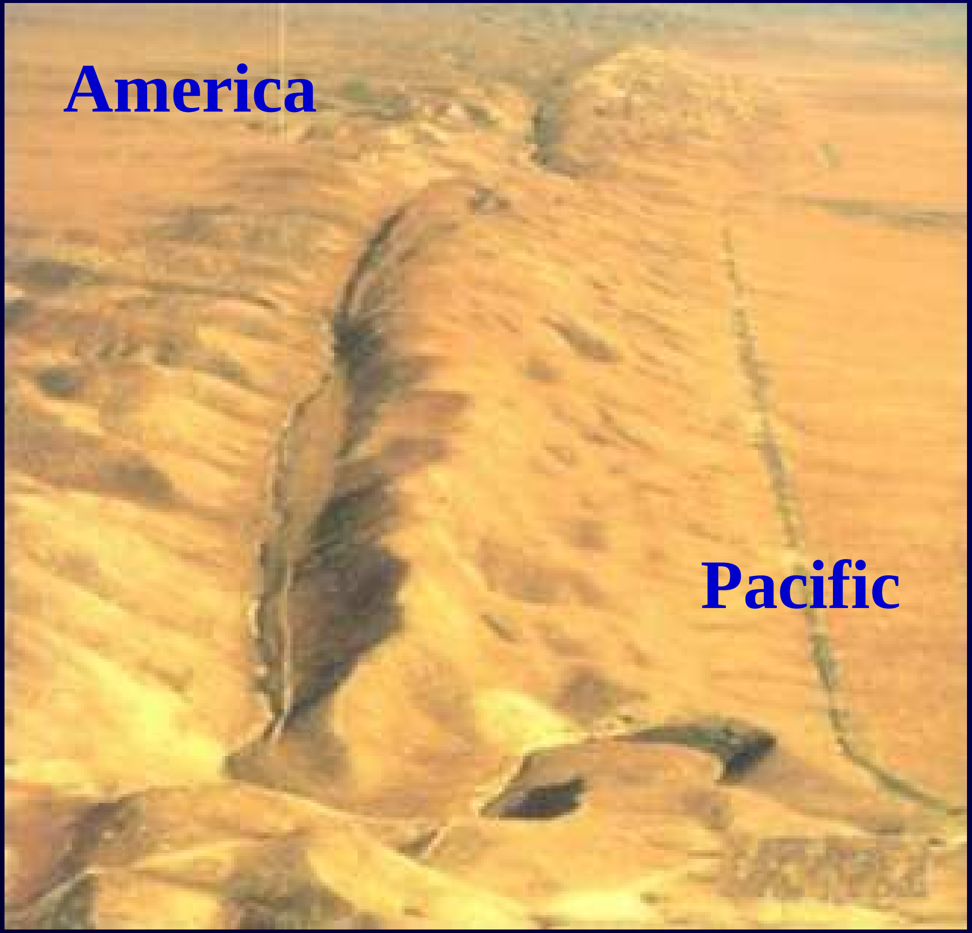


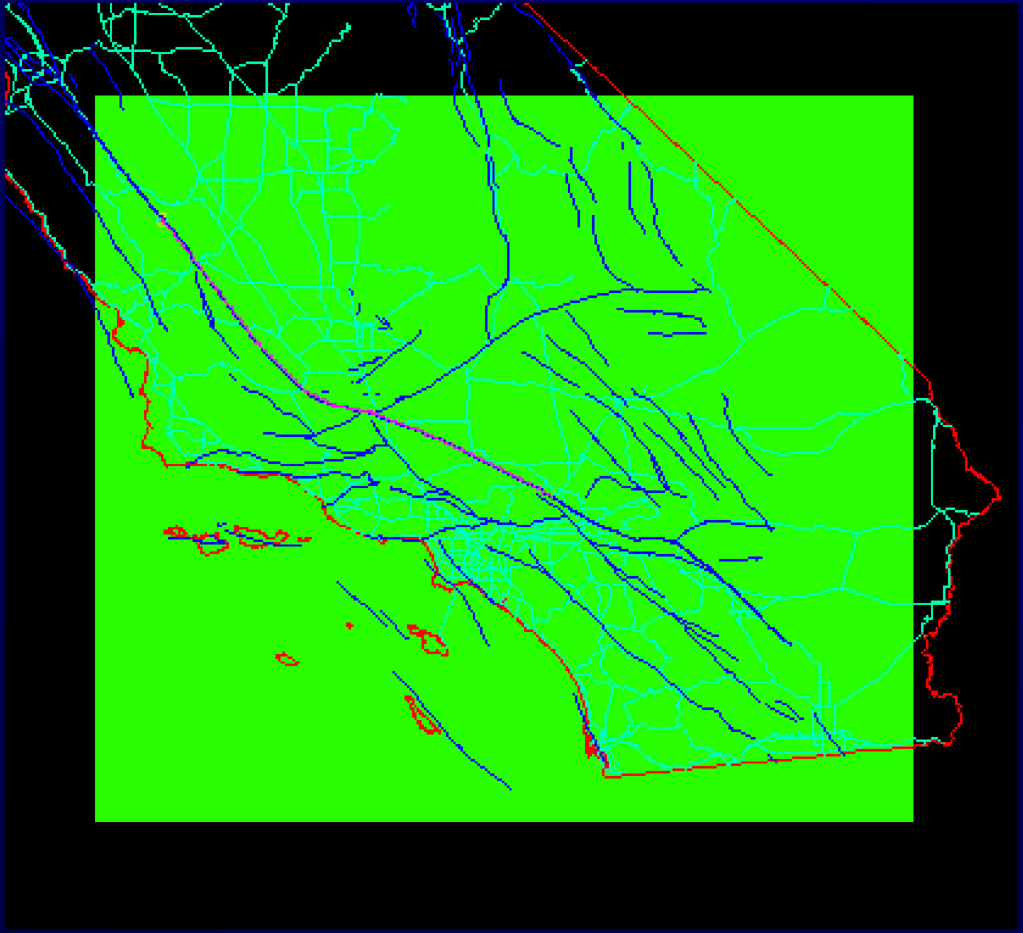
San Andreas fault - Carrizo Plain



Horizontal scale approximately 200 m

San Andreas – January 9, 1857





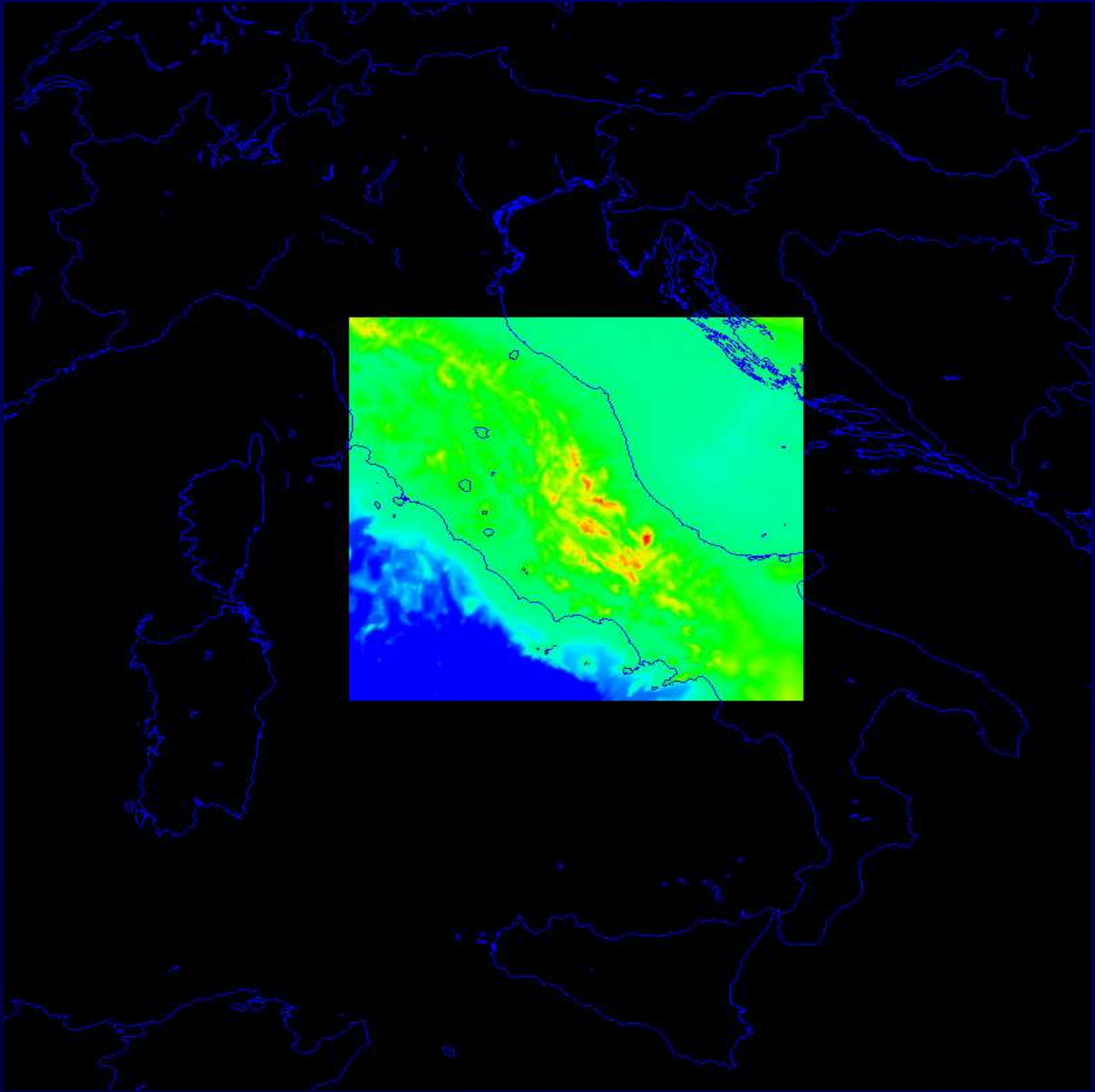
Vertical scale approximately 1 km

Scale approximately 500 km

Carrizo Plain, San Andreas Fault, California, USA

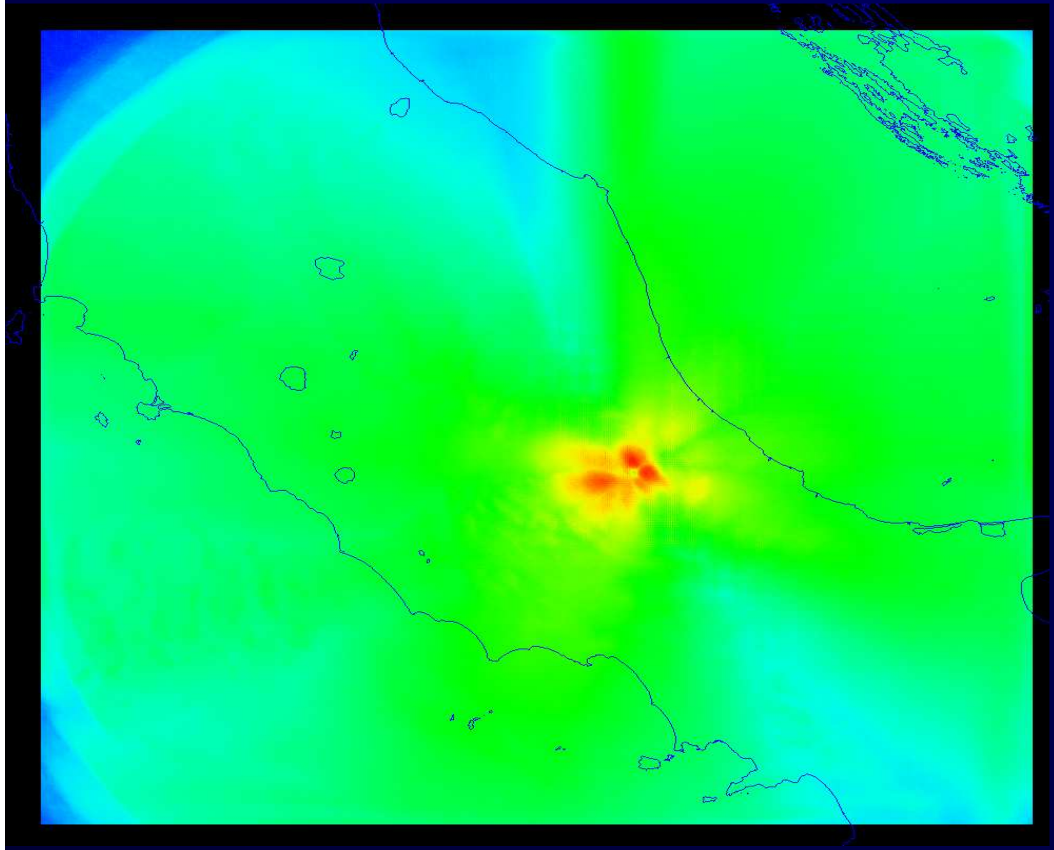
L’Aquila, Italy, April 6, 2009 (Mw = 6.2)

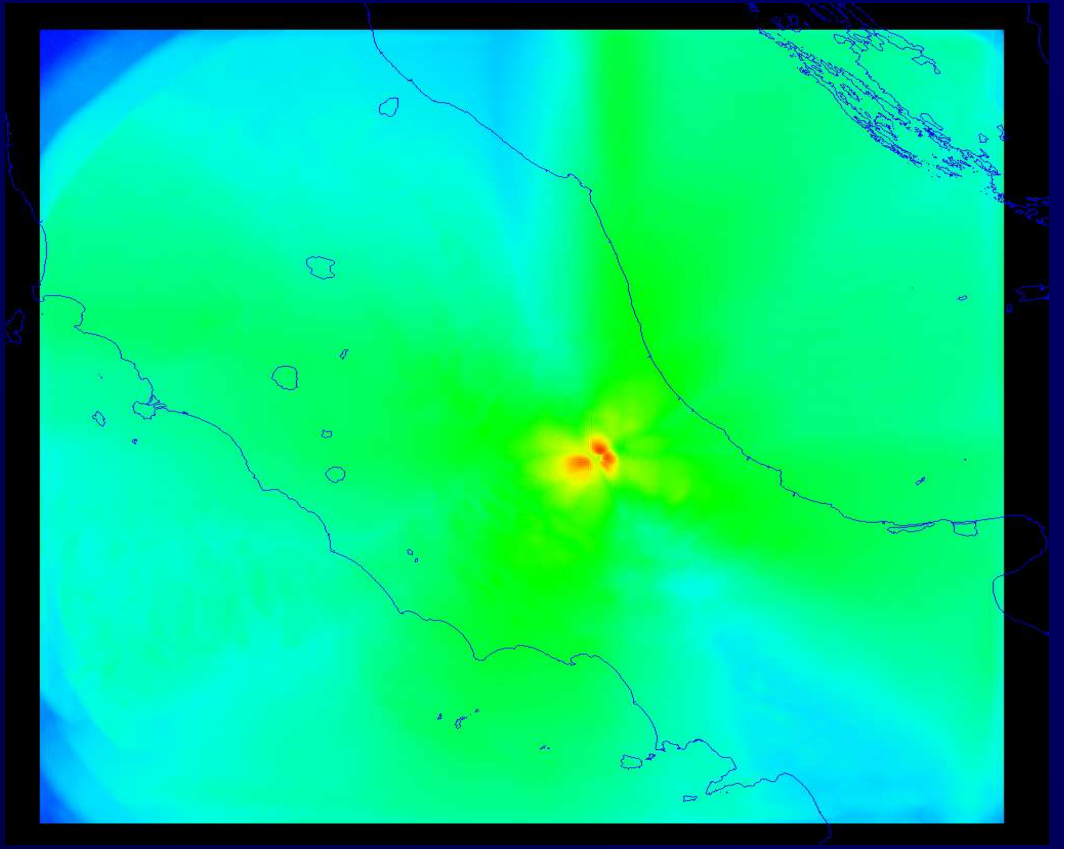
Forlì
Firenze
San Marino
Italia
Italy
Perugia
Ancona
Foligno
Terni
Pescara
L'Aquila
Roma
Latina
Split
Herzegovina
Sarajevo
Mostar
Foggia
Andria
Bari
Monopoli
Napoli
Potenza
Salerno
Taranto
Brindisi
Lecce
Catanzaro
©2009 Google - Données cartographiques ©2009 Tele Atlas - Conditions



Ran on the JADE and TITANE supercomputers in France

L’Aquila, Italy, April 6, 2009 (Mw = 6.2)





Peak groud velocity maps obtained on April 7, 2009, for two hypothetical earthquake scenarios and sent to the local authorities in the evening.

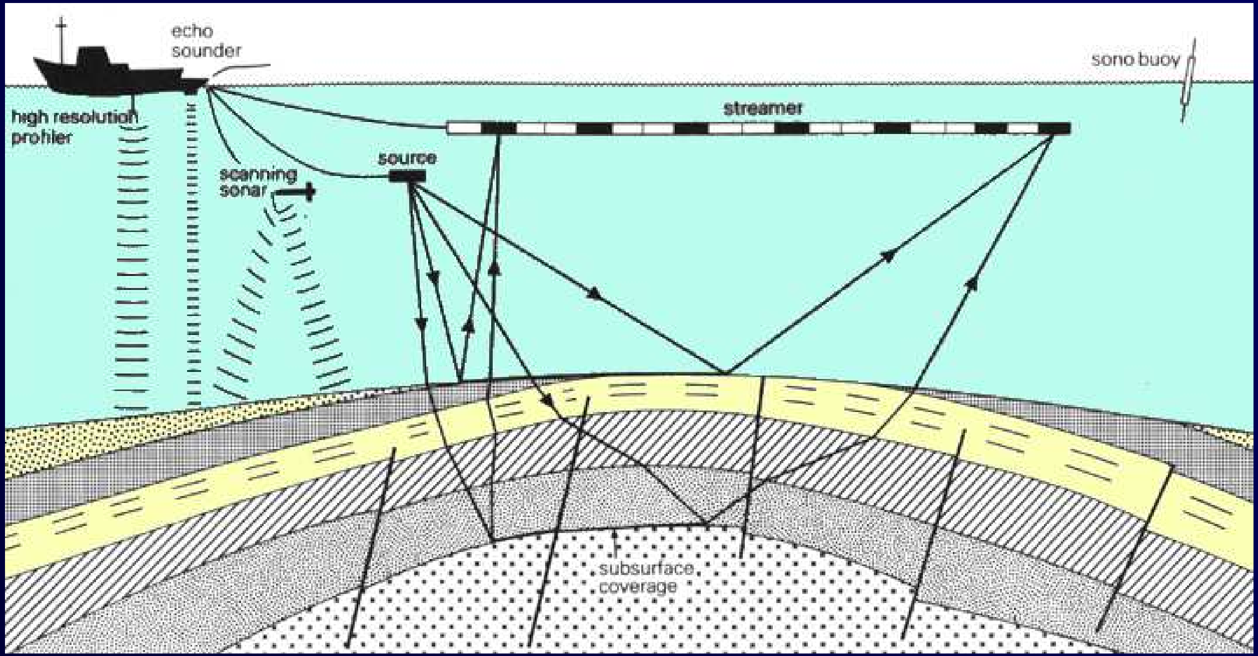
Dec 26, 2004 Sumatra event

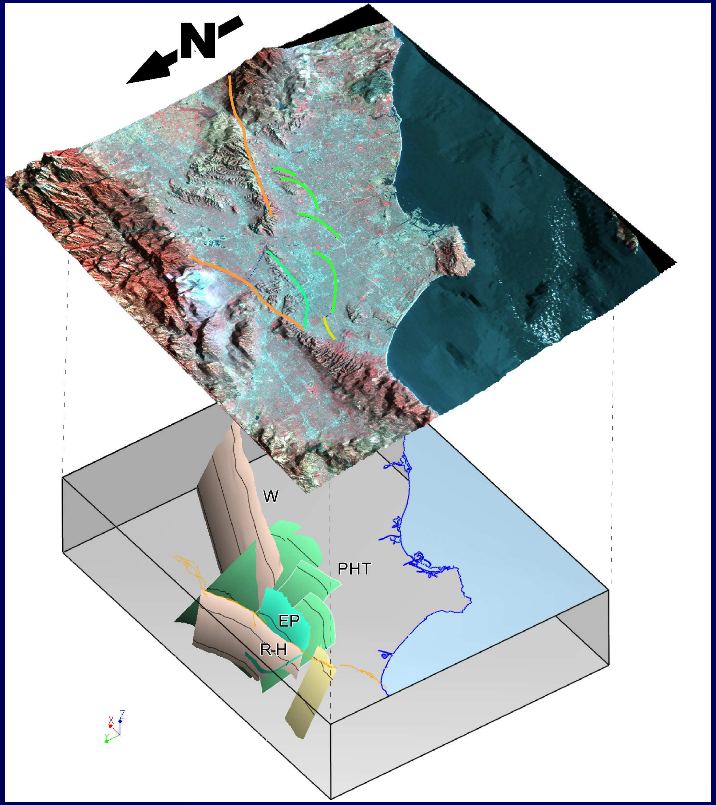
20°
10°
0°
-10°
70°
80°
90°
100°
110°

From Tromp et al., 2005

Vertical component of velocity at periods of 10 s and longer on a regional scale

Collaboration with the oil industry

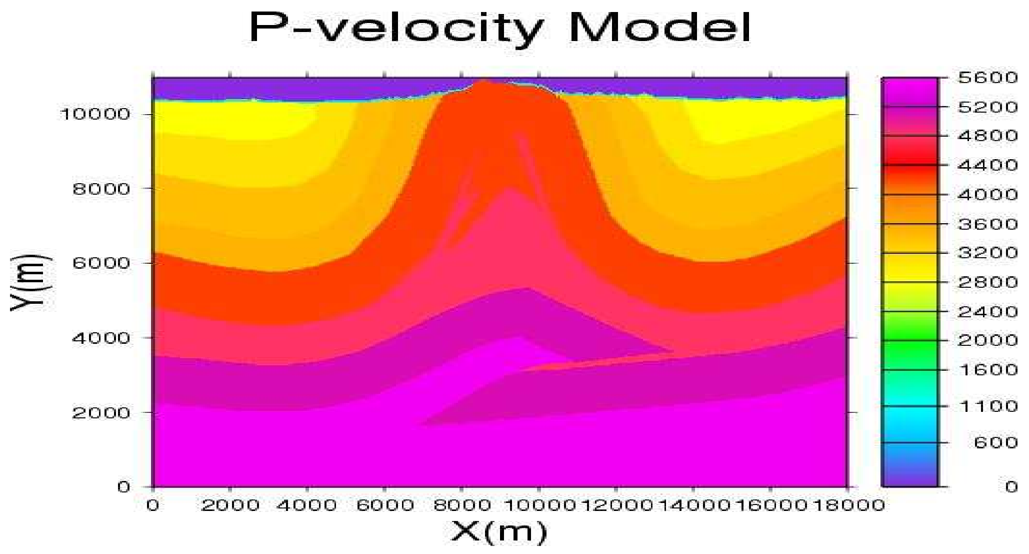


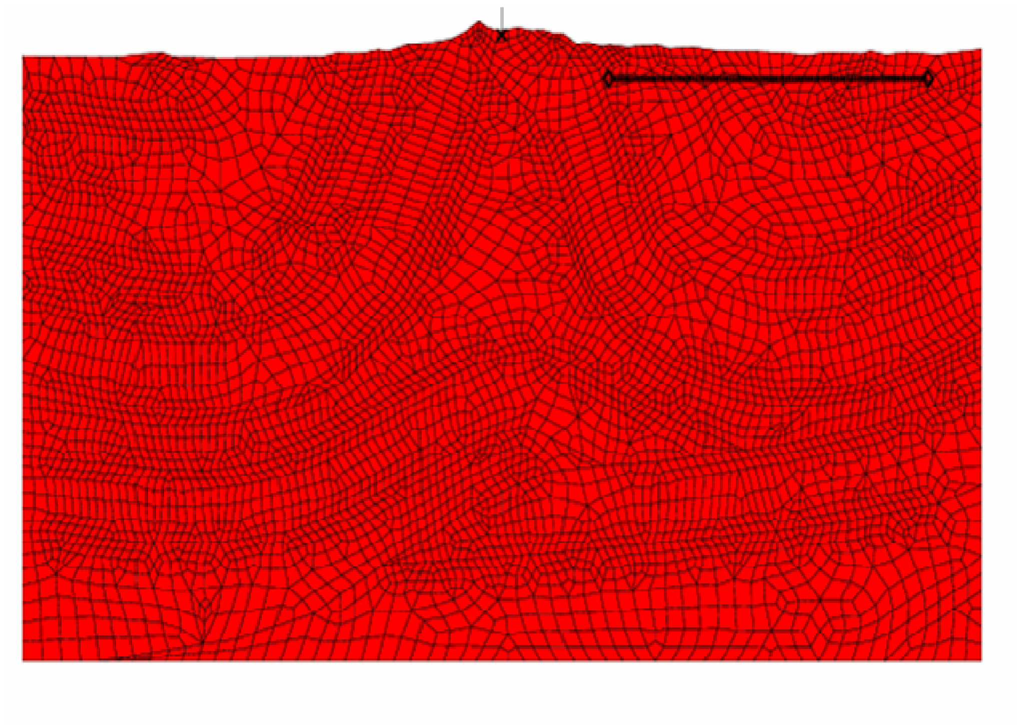


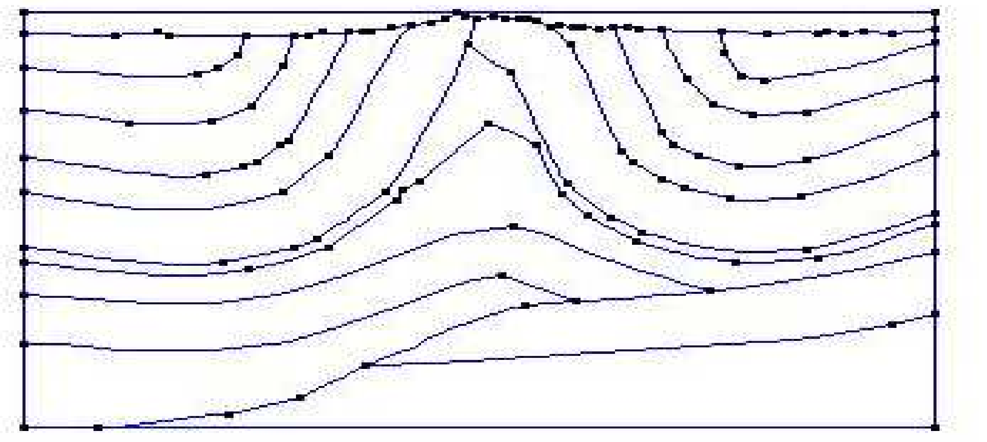
Dynamic geophysical technique of imaging subsurface geologic structures by generating sound waves at a source and recording the reflected components of this energy at receivers.

The seismic data analysis technique is the *industry standard* for locating subsurface oil and gas accumulations.

Meshing an oil industry model







- Structures géologiques dans les Andes (Bolivie)
- Couche fine altérée en surface

→ Problème de dispersion en surface (Freq0 > 10 Hz).

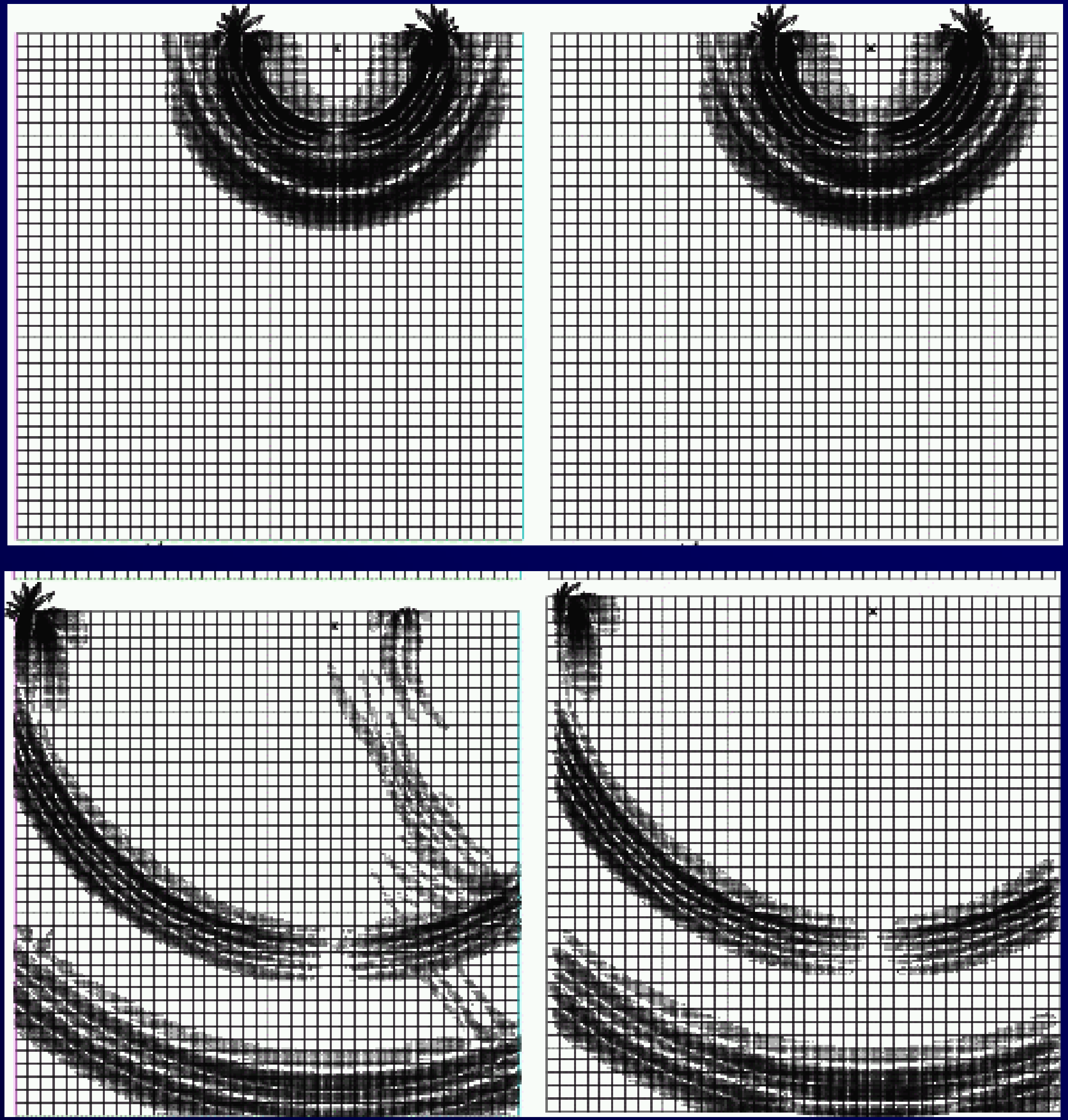
- 5.3 millions de points à 10 Hz.
- Générateur GiD automatique de maillage (UPC/ CIMNE). 98% des angles $45^o < \theta < 135^o$.

Pires angles: 9.5^o and 172^o

Sandrine Fauqueux
Thèse INRIA/IFP (2003)

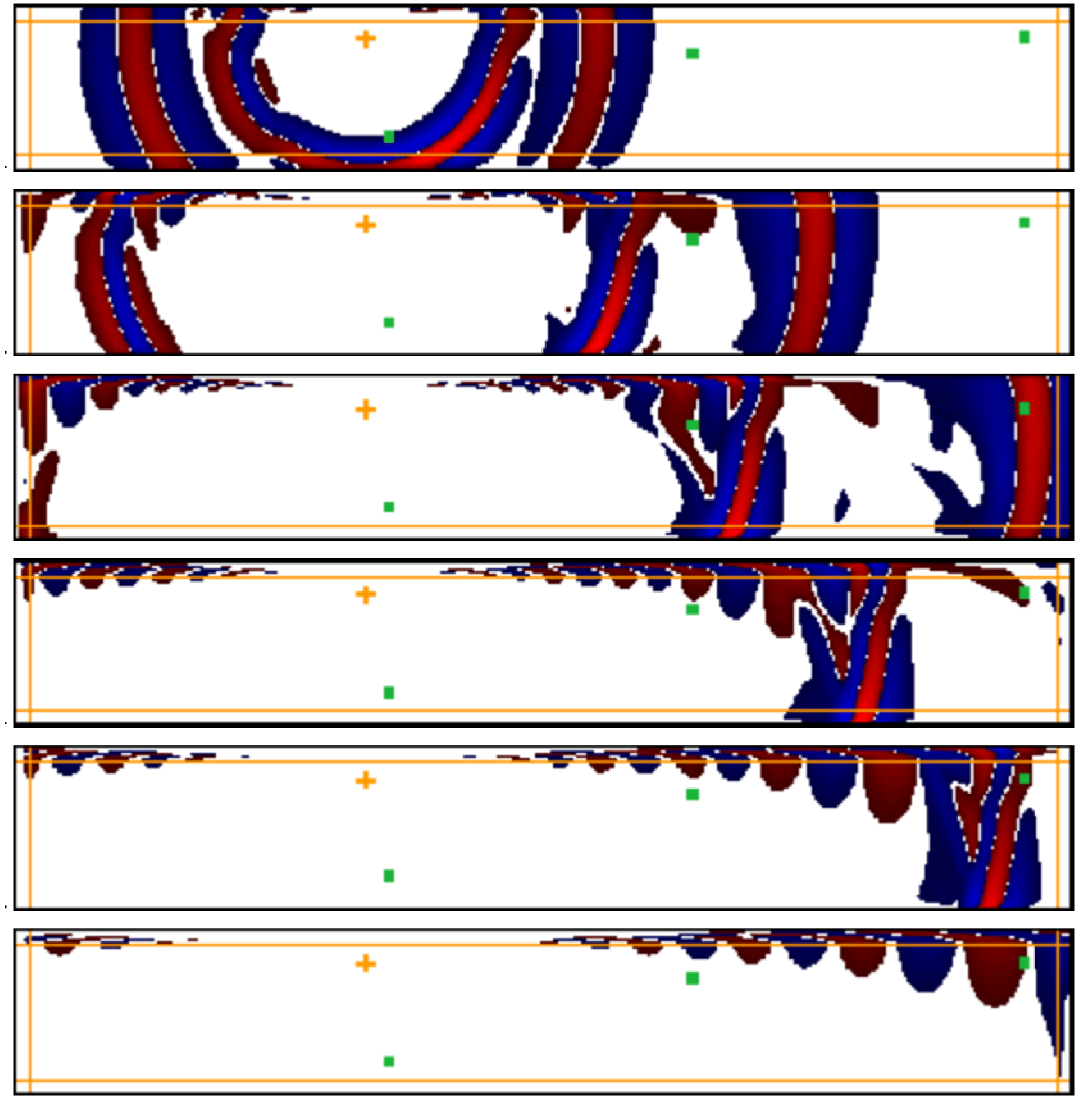
Absorbing conditions

- Used to be a big problem
- Bérenger 1994
- INRIA (Collino, Cohen)
- Extended to second-order systems by Komatitsch and Tromp (2003)



PML (Perfectly Matched Layer) $\Rightarrow$ Hélène Barucq

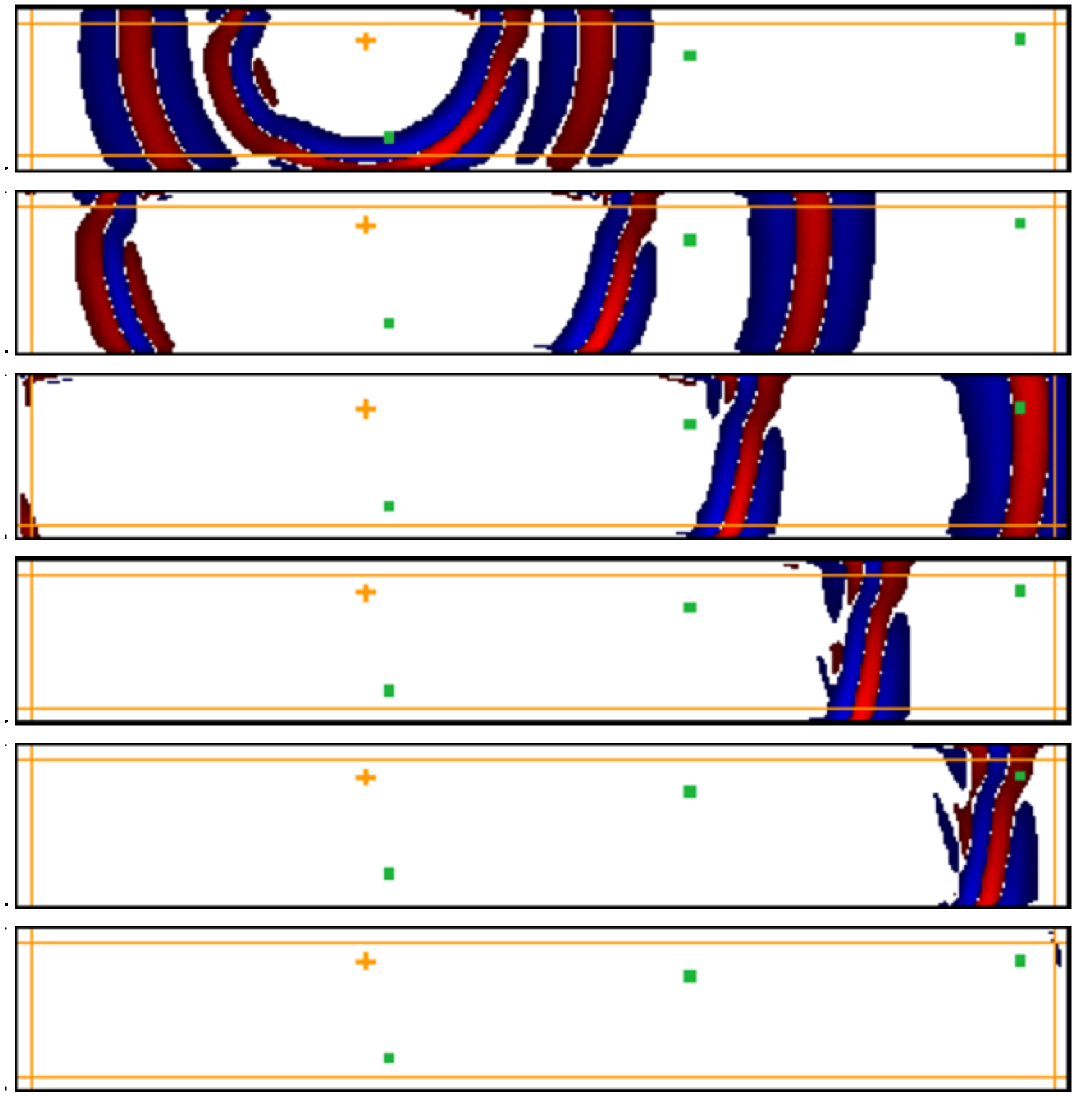
Classical PML in 2D for seismic waves



- Not optimized for grazing incidence
- Usually split
- Produces artefacts of significant amplitude at grazing incidence

Finite-difference technique in velocity and stress: staggered grid of Madariaga (1976), Virieux (1986)

Convolution-PML in 2D for seismic waves



- Optimized for grazing incidence
- *Not* split
- Use recursive convolution based on memory variables (Luebbers and Hunsberger 1992)
- « 3D at the cost of 2D »

Komatitsch and Martin, Geophysics (2007).

Adjoint Method (Waveforms) Tromp et al. (2006, 2008, 2009)

$$\chi_1(\mathbf{m}) = \frac{1}{2}\sum_{r=1}^{N_r}\int_0^T w_r(t)||\mathbf{s}(\mathbf{x}_r,t;\mathbf{m}) - \mathbf{d}(\mathbf{x}_r,t)||^2\,\mathrm{d}t,$$

$$\delta\chi_1 = \int_V [\boxed{K_\rho(\mathbf{x})}\,\delta\ln\rho(\mathbf{x}) + \boxed{K_\mu(\mathbf{x})}\,\delta\ln\mu(\mathbf{x}) + \boxed{K_\kappa(\mathbf{x})}\,\delta\ln\kappa(\mathbf{x})]\,\mathrm{d}^3\mathbf{x},$$

$$K_\rho(\mathbf{x}) = -\int_0^T \rho(\mathbf{x})\,\partial_t\mathbf{s}^\dagger(\mathbf{x},T-t)\cdot\partial_t\mathbf{s}(\mathbf{x},t)\,\mathrm{d}t,$$

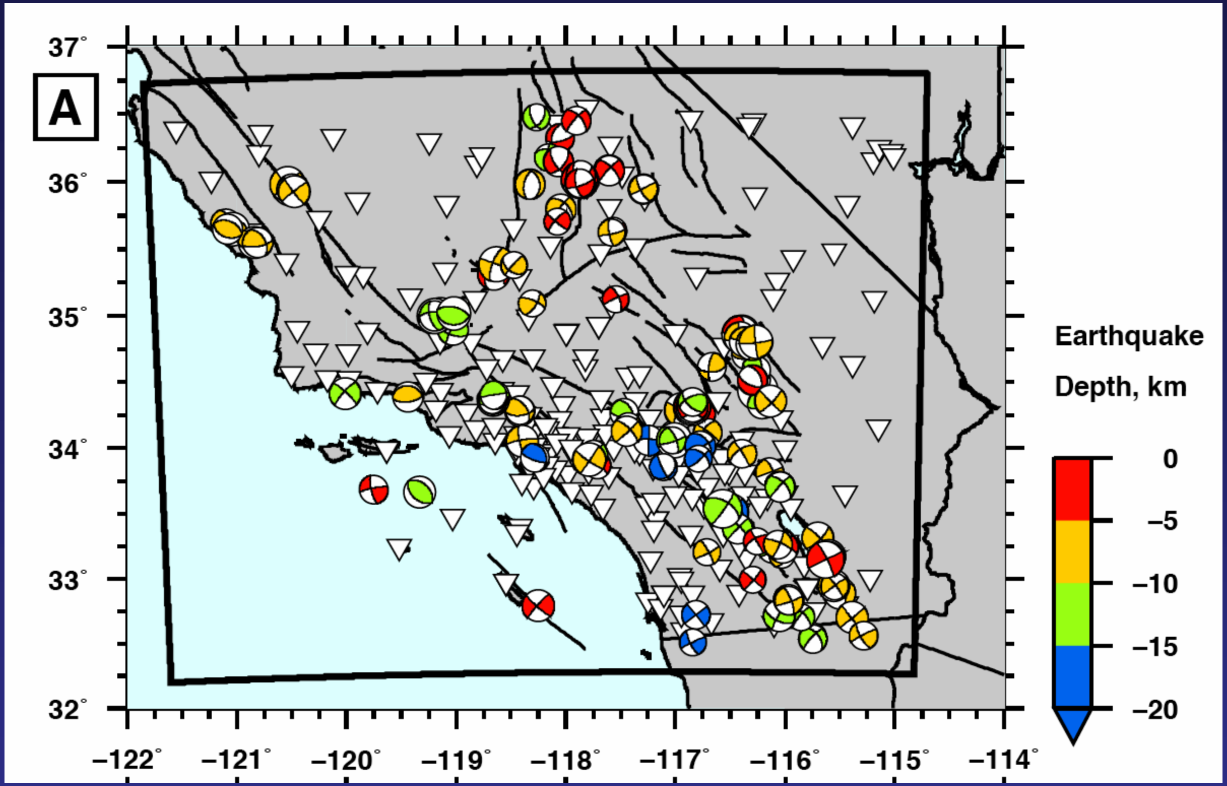
$$K_\mu(\mathbf{x}) = -\int_0^T 2\mu(\mathbf{x})\,\mathbf{D}^\dagger(\mathbf{x},T-t):\mathbf{D}(\mathbf{x},t)\,\mathrm{d}t,$$

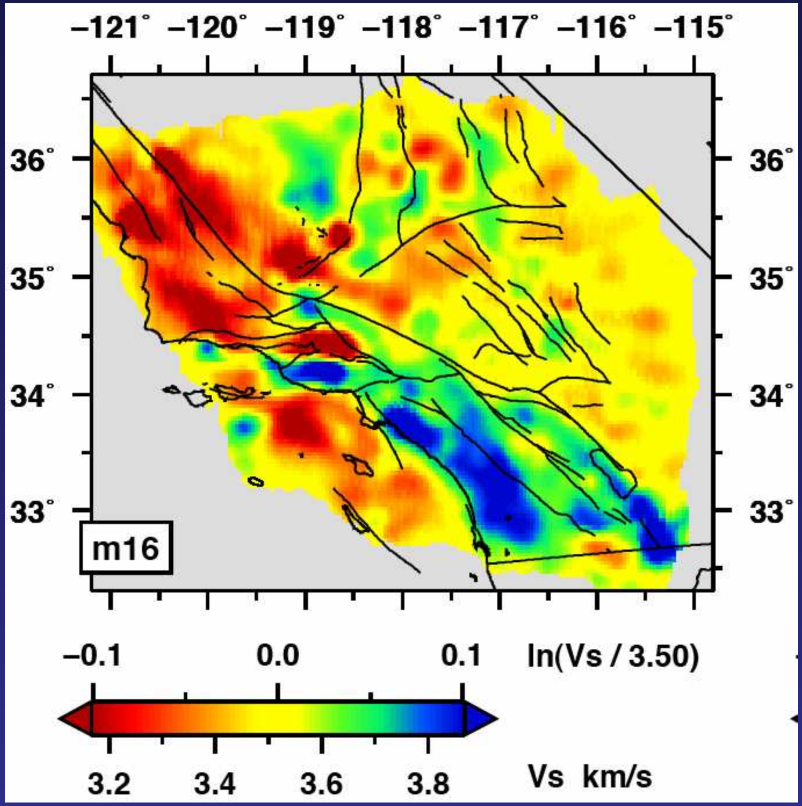
Princeton Univ

$$K_\kappa(\mathbf{x}) = -\int_0^T \kappa(\mathbf{x})\,[\nabla\cdot\mathbf{s}^\dagger(\mathbf{x},T-t)]\,[\nabla\cdot\mathbf{s}(\mathbf{x},t)]\,\mathrm{d}t,$$

$$\mathbf{f}_1^\dagger(\mathbf{x},t) = \sum_{r=1}^{N_r} w_r(t)[\mathbf{s}(\mathbf{x}_r,T-t) - \mathbf{d}(\mathbf{x}_r,T-t)]\,\delta(\mathbf{x}-\mathbf{x}_r),$$

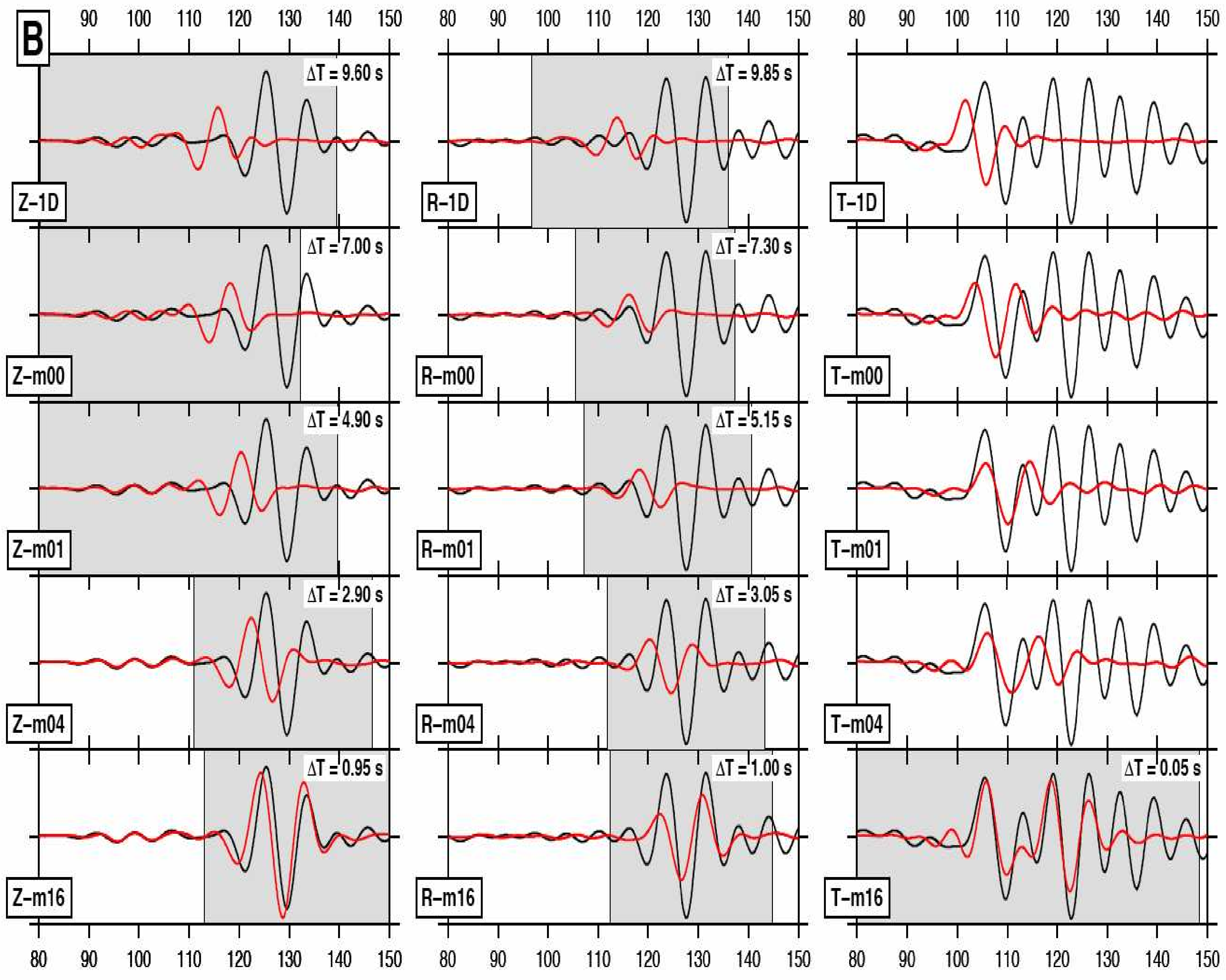
Tape et al. (2009): 143 earthquakes used in inversion





Depth 10 km

- 3 simulations per earthquake per iteration
- 16 iterations
- 6,864 simulations
- 168 processor cores per simulation
- 45 minutes of wall-clock time per simulation
- 864,864 processor core hours


B
ΔT = 9.60 s
Z–1D
ΔT = 9.85 s
R–1D
T–1D
ΔT = 7.00 s
Z–m00
ΔT = 7.30 s
R–m00
T–m00
ΔT = 4.90 s
Z–m01
ΔT = 5.15 s
R–m01
T–m01
ΔT = 2.90 s
Z–m04
ΔT = 3.05 s
R–m04
T–m04
ΔT = 0.95 s
Z–m16
ΔT = 1.00 s
R–m16
ΔT = 0.05 s
T–m16
80 90 100 110 120 130 140 150

Conclusions and future work

Dimitri Komatitsch, David Michéa and Gordon Erlebacher, Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA, Journal of Parallel and Distributed Computing, vol. 69(5), p. 451-460, doi: 10.1016/j.jpdc.2009.01.006 (2009).

Dimitri Komatitsch, Gordon Erlebacher, Dominik Göddeke and David Michéa, High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster, Journal of Computational Physics, vol. 229(20), p. 7692-7714, doi: 10.1016/j.jcp.2010.06.024 (2010).

- CUDA on a single GPU leads to a speedup of 25x for our application versus a single CPU core
- We keep a speedup of 20x when we use a cluster of GPUs with non-blocking MPI (12x if we compare to all the CPU cores)
- But code development is currently invasive and time consuming
- In future work, we could use OpenCL
- Need for some kind of OpenMP for GPUs: CAPS HMPP, StarSs, StarPU...