Combining recent HPC techniques for 3D geophysics acceleration

3^{rd} HOSCAR workshop, 02-05 September 2013, Bordeaux (France)

- Lionel BOILLOT, PhD student – INRIA Magique3D
- Emmanuel AGULLO, Runtime researcher – INRIA Hiepacs
- George BOSILCA, Runtime researcher – ICL Univ. of Tennesse
- Henri CALANDRA, PhD supervisor – TOTAL

Starting point

Objective

Speed up the code with HPC techniques

Massively-parallel code based on the MPI library.

Industrial constraints

- scalability
- readability
- portability

Bonus: non-intrusive

Propositions

- automatic parallelism over runtimes
- utilization of coprocessors through the runtime

Table of contents

1. Physics & Maths background
   - Geophysics context
   - Numerical scheme
2. Preliminary work
3. Runtimes
   - Description
   - DAG of DIP
   - Very first results

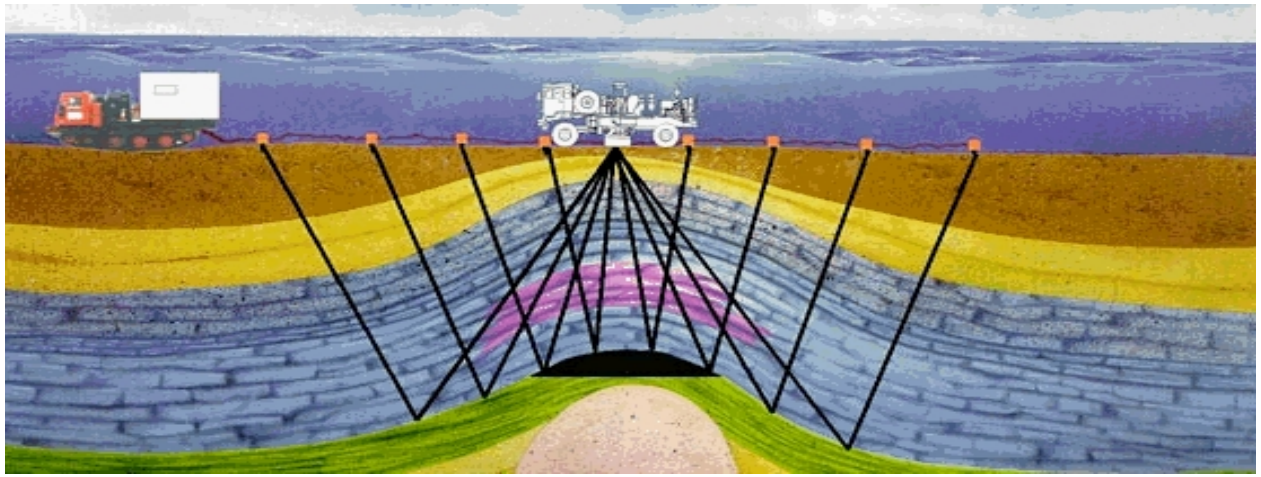
Outline

1. Physics & Maths background
   - Geophysics context
   - Numerical scheme
2. Preliminary work
3. Runtimes
   - Description
   - DAG of DIP
   - Very first results

RTM context

Geophysics:

- Hydrocarbons detection: petroleum or natural gas
- Earth medium: seismic waves, heterogeneous complex domain



Simulation:

- Seismic imaging: find the subsurface layers
- Equations: elastic/acoustic wave in 2D/3D

Reverse Time Migration (RTM)

Iterative method based on multiple wave equation resolutions

Example: RTM in acoustics

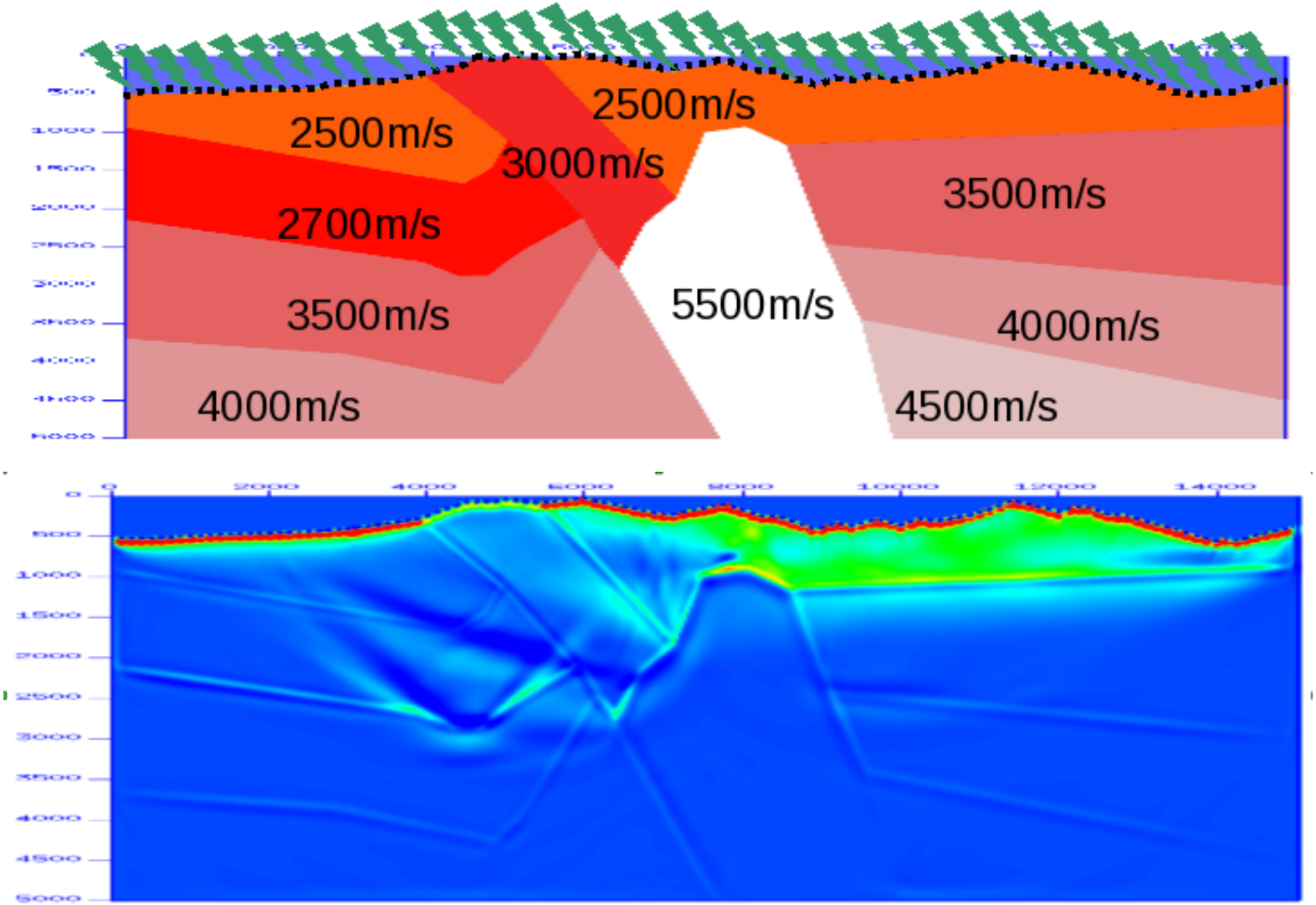


Figure: RTM in acoustic

Elastic wave equation

Let us consider:

- Ω, an open bounded domain of $\mathbb{R}^2$
- $\Gamma = \partial\Omega$, the domain boundary; $\mathbf{n}_\Gamma$, the exterior normal
- $\mathbf{x} = (x, z) \in \Omega$ and $t \in [0, T]$, the space and time variables

Velocity-stress formulation

$$\begin{cases} \rho(\mathbf{x})\partial_t v(\mathbf{x}, t) &= \nabla . \underline{\underline{\sigma}}(\mathbf{x}, t) \\ \partial_t \underline{\underline{\sigma}}(\mathbf{x}, t) &= \underline{\underline{C}}(\mathbf{x}) : \underline{\underline{\epsilon}}(v(\mathbf{x}, t)) \end{cases} \tag{1}$$

- $\rho > 0$, the density
- $v \in \mathbf{H}^1(\Omega \times [0, T])$, the unknown velocity field
- $\underline{\underline{\sigma}} \in \underline{\underline{H}}_{div}(\Omega \times [0, T])$, the stress tensor
- $\underline{\underline{C}}$, the stiffness tensor (elasticity tensor)
- $\underline{\underline{\epsilon}}(v) = \frac{1}{2}(\boldsymbol{\nabla} v + (\boldsymbol{\nabla} v)^T)$, the strain tensor

Space discretization 1/2

Ω_h: polygonal mesh (partition of Ω), composed of n_K elements K, with Γ_{in}, the internal boundaries and Γ_{out}, the external boundaries.

Discontinuous Galerkin Method (DGM)

- Approximate v and $\underline{\underline{\sigma}}$ by discontinuous functions such that $\{v_h, \underline{\underline{\sigma}}_h\} \in L^2(\Omega_h \times [0,T])$ with $\{v_h|_K, \underline{\underline{\sigma}}_h|_K\} \in H^1(K \times [0,T])\ \forall K$.
- Integrate against discontinuous test functions w and $\underline{\underline{\xi}}$.

Integration on K

$$\begin{cases} \int_K \rho_K \partial_t v w d\mathbf{x} &= \int_{\partial_K} \underline{\underline{\sigma}}\mathbf{n_K}.w d\mathbf{x} - \int_K \underline{\underline{\sigma}} : \boldsymbol{\nabla} w d\mathbf{x} \\ \int_K \partial_t \underline{\underline{\sigma}} : \underline{\underline{\xi}} d\mathbf{x} &= \int_{\partial_K} (\underline{\underline{C}} : \underline{\underline{\xi}})\mathbf{n_K}.v d\mathbf{x} - \int_K v.\nabla.(\underline{\underline{C}} : \underline{\underline{\xi}}) d\mathbf{x} \end{cases} \quad (2)$$

Space discretization 2/2

Let us consider $\overline{\Gamma} = \Gamma_{in} \cup \Gamma_{out}$.

$$\begin{cases} \sum_K \int_K \rho_K \partial_t v w d\mathbf{x} = \\ \qquad \sum_{\overline{\Gamma}} \int_{\overline{\Gamma}} \{\!\{ \underline{\underline{\sigma}} \mathbf{n_K} \}\!\}.[\![w]\!] d\mathbf{x} - \sum_K \int_K \underline{\underline{\sigma}} : \boldsymbol{\nabla} w d\mathbf{x} \\ \sum_K \int_K \partial_t \underline{\underline{\sigma}} : \underline{\underline{\xi}} d\mathbf{x} = \\ \qquad \sum_{\overline{\Gamma}} \int_{\overline{\Gamma}} [\![(\underline{\underline{C}} : \underline{\underline{\xi}}) \mathbf{n_K}]\!].\{\!\{ v \}\!\} d\mathbf{x} - \sum_K \int_K v.\nabla.(\underline{\underline{C}} : \underline{\underline{\xi}}) d\mathbf{x} \end{cases} \tag{3}$$

Semi-discrete form

$$\begin{cases} M_v \partial_t v_h + R_{\underline{\underline{\sigma}}} \underline{\underline{\sigma}}_h &= 0 \\ M_{\underline{\underline{\sigma}}} \partial_t \underline{\underline{\sigma}}_h + R_v v_h &= 0 \end{cases} \tag{4}$$

Time discretization

Time domain $[0, T]$ divided into time steps Δt:

Leap-frog time scheme

Iteration on n:

$$\begin{cases} M_v \dfrac{v_h^{n+1} - v_h^n}{\Delta t} + R_{\underline{\underline{\sigma}}} \underline{\underline{\sigma}}_h^{n+1/2} & = 0 \\ M_{\underline{\underline{\sigma}}} \dfrac{\underline{\underline{\sigma}}_h^{n+3/2} - \underline{\underline{\sigma}}_h^{n+1/2}}{\Delta t} + R_v v_h^{n+1} & = 0 \end{cases} \tag{5}$$

M_v and $M_{\underline{\underline{\sigma}}}$ block-diagonal matrices $\Rightarrow$ quasi-explicit scheme!

$\Rightarrow$ Allow a massively parallel simulation algorithm

Very simple DIP Algorithm

$$\begin{cases} M_v \dfrac{v_h^{n+1} - v_h^n}{\Delta t} + R_{\underline{\underline{\sigma}}} \underline{\underline{\sigma}}_h^{n+1/2} & = 0 \\ \\ M_{\underline{\underline{\sigma}}} \dfrac{\underline{\underline{\sigma}}_h^{n+3/2} - \underline{\underline{\sigma}}_h^{n+1/2}}{\Delta t} + R_v v_h^{n+1} & = 0 \end{cases} \tag{6}$$

$$\begin{cases} v_h^{n+1} & = v_h^n + M_v^{-1}[\Delta t R_{\underline{\underline{\sigma}}} \underline{\underline{\sigma}}_h^{n+1/2}] \\ \underline{\underline{\sigma}}_h^{n+3/2} & = \underline{\underline{\sigma}}_h^{n+1/2} + M_{\underline{\underline{\sigma}}}^{-1}[\Delta t R_v v_h^{n+1}] \end{cases} \tag{7}$$

Initialization($mesh, matrix, v, \sigma$)
For $n = 1 : n\_timesteps\_T$
$v_h^{n+1} \leftarrow computeVelocity(v_h^n, \sigma_h^{n+1/2}, \Delta_t)$
$\sigma_h^{n+3/2} \leftarrow computeStress(\sigma_h^{n+1/2}, v_h^{n+1}, \Delta_t)$
End For t

Outline

1. Physics & Maths background
   - Geophysics context
   - Numerical scheme
2. Preliminary work
3. Runtimes
   - Description
   - DAG of DIP
   - Very first results

Profile and optimize

DIP code profiling revealed hotspots:

Sequential profiling

- minimize cache misses
- check unrolling loops
- use vectorization

Parallel profiling

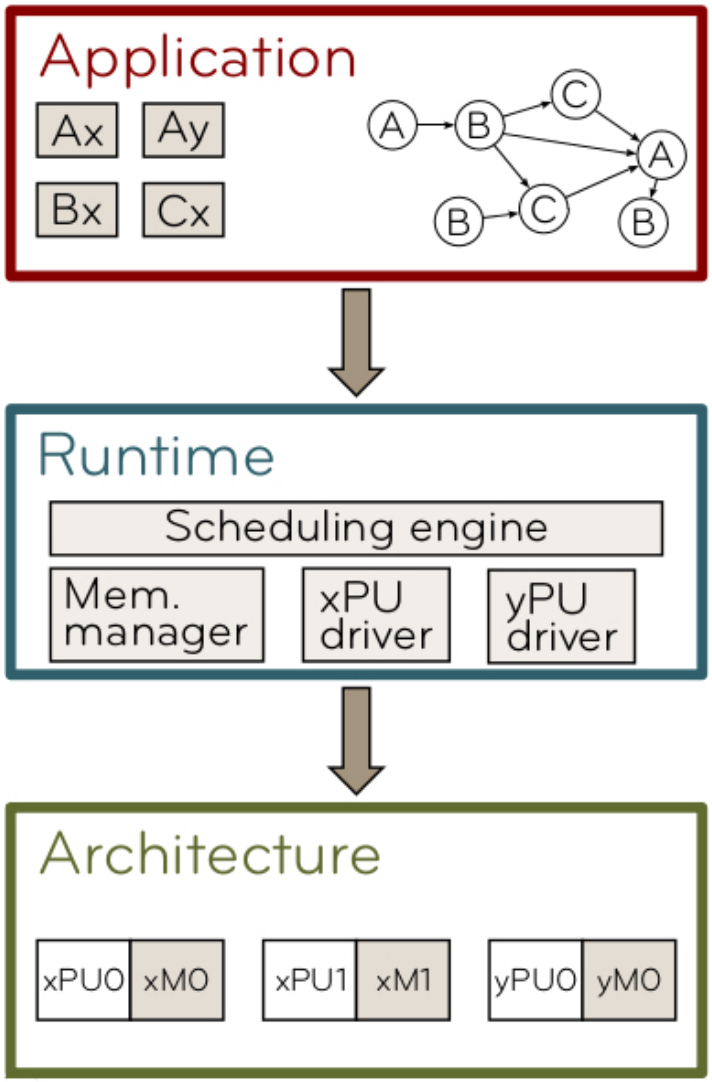
- avoid global barriers
- minimize the communications
- check the partitionning load-balancing

$\Rightarrow$ first optimization steps bring two-fold acceleration!

Outline

1. Physics & Maths background
   - Geophysics context
   - Numerical scheme
2. Preliminary work
3. Runtimes
   - Description
   - DAG of DIP
   - Very first results

Basic scheme



Courtesy of Alfredo Buttari

Runtime:

- abstraction layer
- hiding heterogeneity

Scheduler:

- where to execute
- when to execute

Memory:

- does the transfert
- guarantees consistency

Who is using runtimes?

Widely adopted in dense linear algebra libraries:

- PLASMA (QUARK)
- DPLASMA (PaRSEC)
- MAGMA-MORSE (StarPU)
- FLAME (SuperMatrix)

but more difficult to use in general complex and irregular workloads...

PhD subject

Use a runtime (PaRSEC) in an industrial-oriented code (DIP)

Special issues:

- unstructured meshes
- not BLAS-based kernels
- load-balancing (physics)

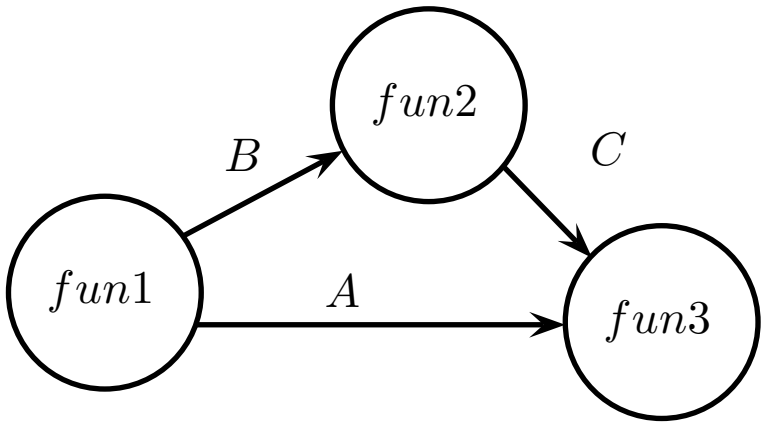
How to use runtimes?

fun1(A: inout, B: out)
fun2(B: in, C:out)
fun3(A: inout, C: in)

Task-based

- Describe the functions as tasks
- Form the dataflow: specify dependancies

$\Rightarrow$ creation of the DAG (Direct Acyclic Graph)



DIP algorithm

For $n = 1 : n\_timesteps\_T$

Communication($\sigma_h^{n+1/2}$)

$v_h^{n+1} \leftarrow computeVelocity(v_h^n, \sigma_h^{n+1/2}, \Delta_t)$

Communication(v_h^{n+1})

$\sigma_h^{n+3/2} \leftarrow computeStress(\sigma_h^{n+1/2}, v_h^{n+1}, \Delta_t)$

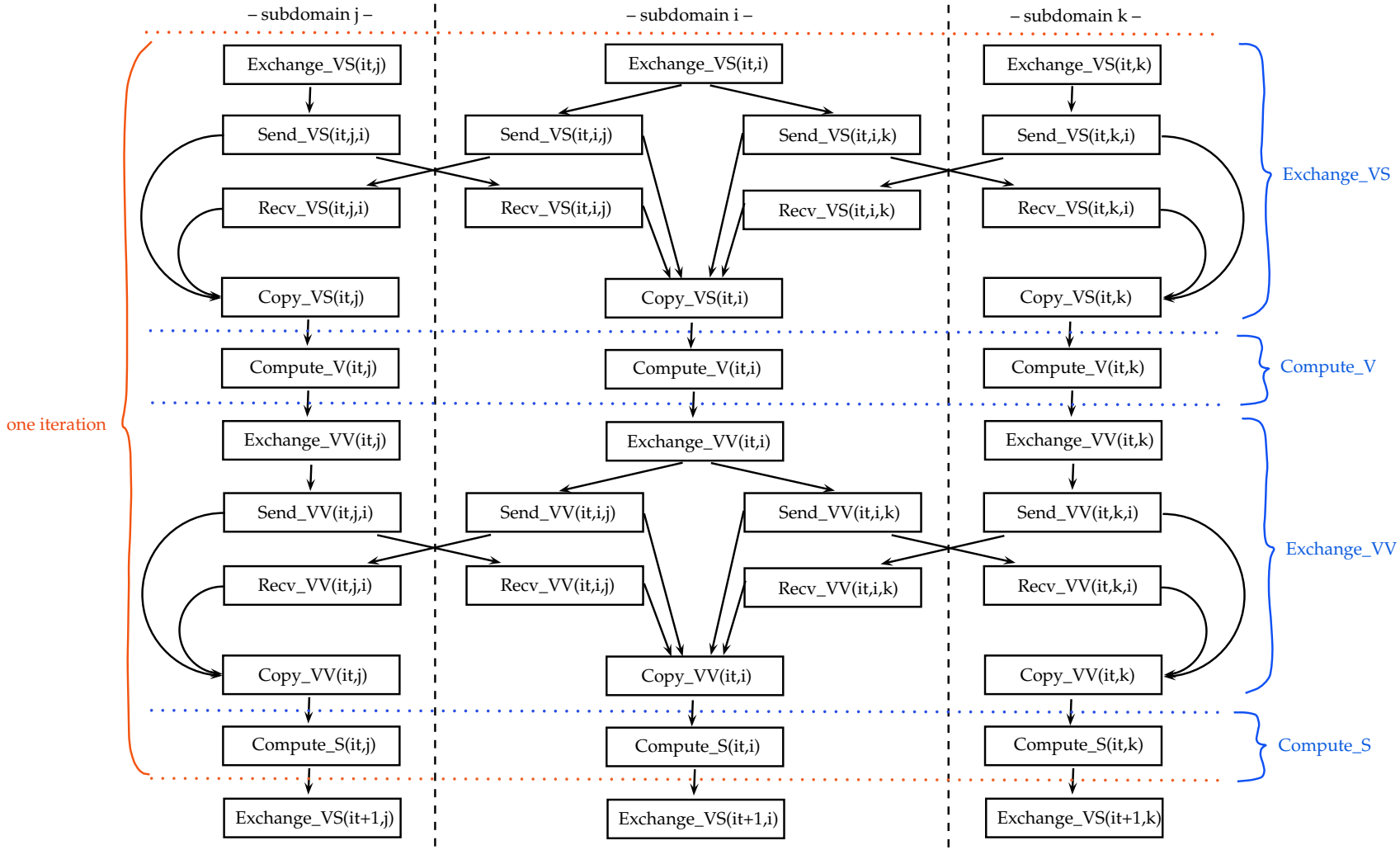
End For t

Let's rename the algorithm steps:

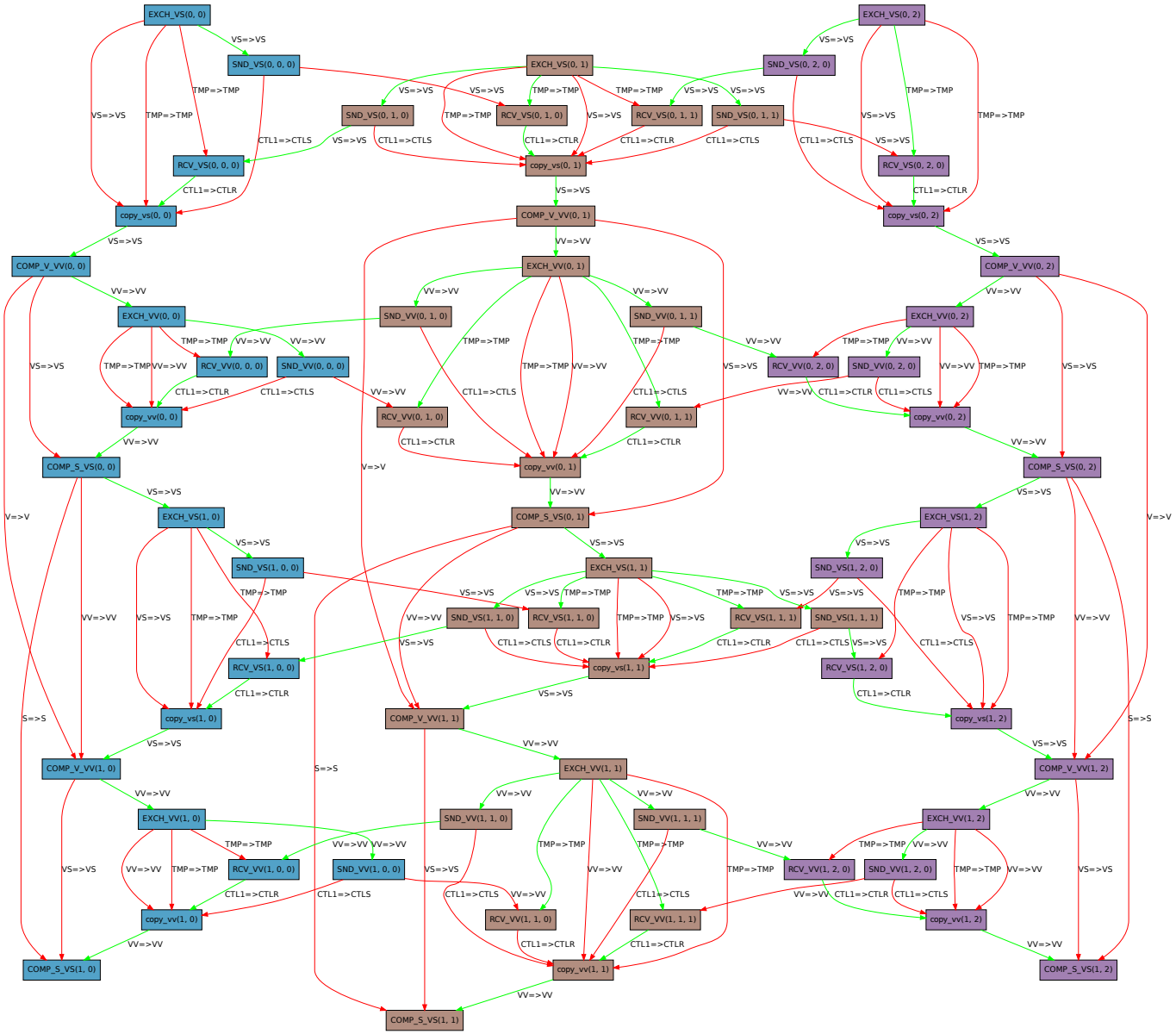
- EXCHANGE_VS
- COMPUTE_V
- EXCHANGE_VV
- COMPUTE_S

Let's divide the EXCHANGE task into SEND, RECV and COPY tasks

DAG of tasks for runtime



PaRSEC DAG display

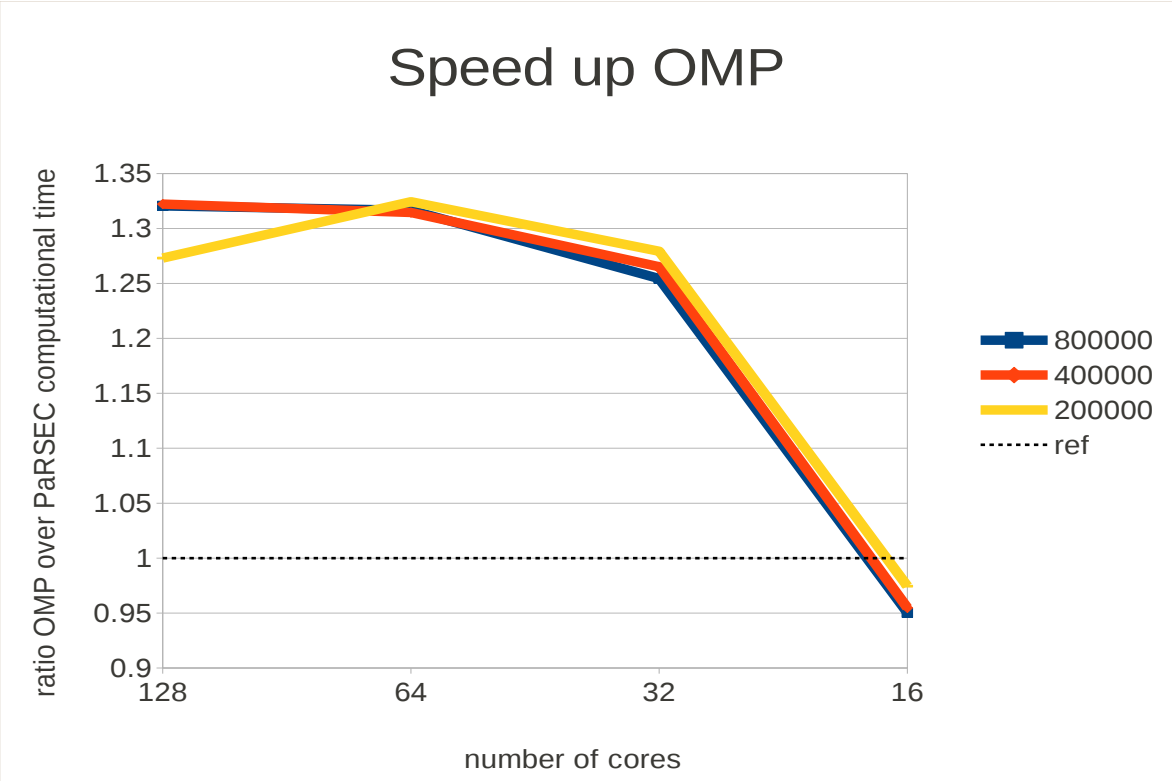


Plafrim shared machine

- PlaFRIM (Inria - Bordeaux) shared memory machine
- 1 SMP node of 160 cores – 20 Intel Xeon CPUs of 8 cores
- NUMA policy (distance from 10 to 55)

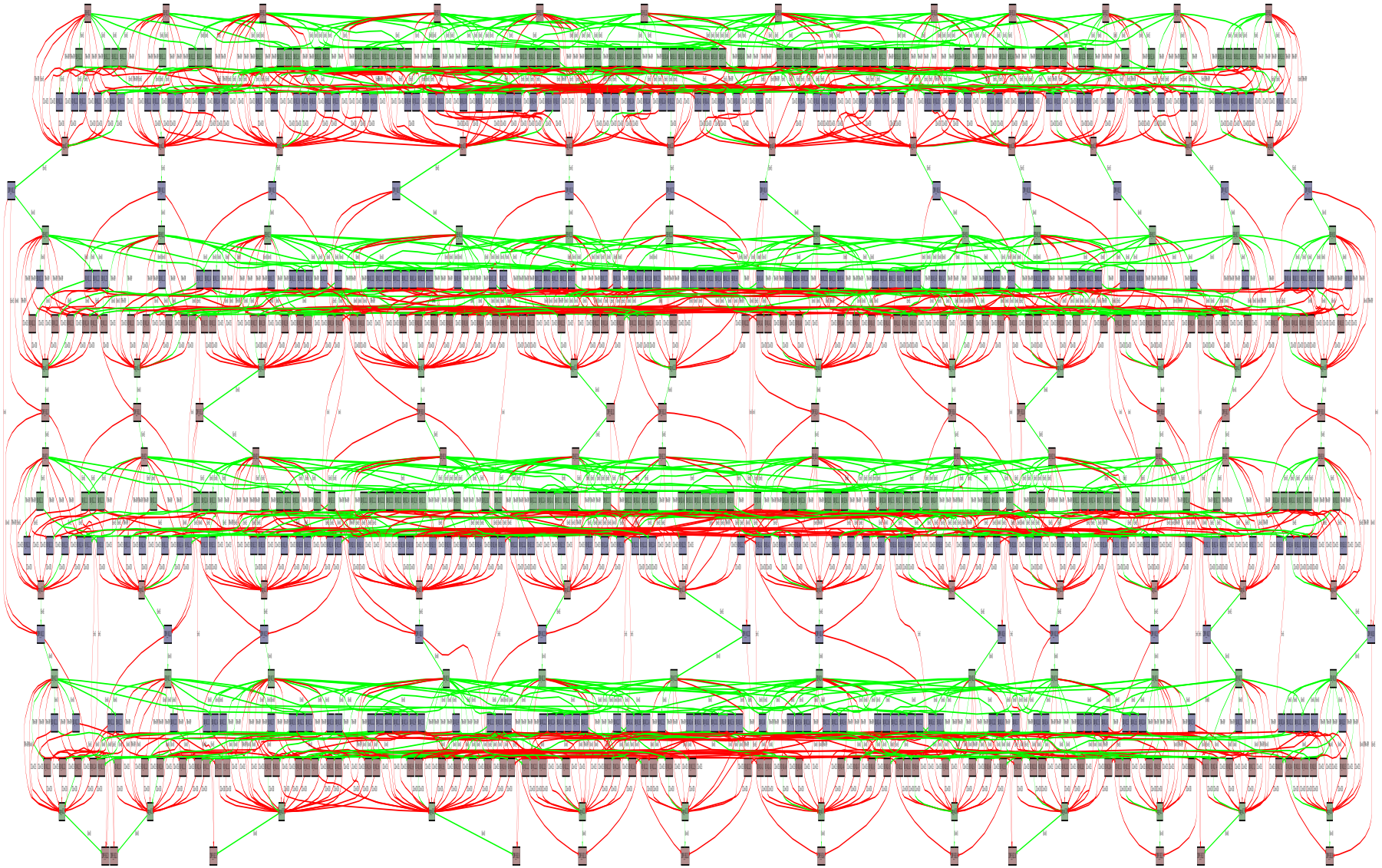
- 3D anisotropic elastic wave equation (highest cost case)
- homogeneous one layer domain (well-balancing)
- Dirichlet bondary condition (well-balancing)

Shared memory results



$\Rightarrow$ 30% of increase in large number of CPUs cases (with granularity)

PaRSEC DAG granularity



Conclusion and perspectives

Conclusion:

- Preliminary optimizations
- Task-based description
- PaRSEC DAG of DIP

Current step

PaRSEC in shared memory $\Rightarrow$ profiling & scalability

Perspectives:

- PaRSEC in distributed memory
- Coprocessors Intel Xeon Phi (MIC)

Support by INRIA-TOTAL strategic action DIP (`http://dip.inria.fr`)