Algorithm and software design for conservation laws

Philippe HELLUY [1,2]

[1]IRMA, Université de Strasbourg, [2]Inria Tonus, France

Forum ORAP, November 2015, Paris

Outlines

Conservation laws

Implicit kinetic schemes

StarPU parallelization

1) Conservation laws

Don't repeat yourself. Use the same framework for several applications.

Conservation laws

Many equations in physics are systems of conservation laws:

$$\partial_t W + \sum_{i=1}^{d} \partial_i F^i(W) = 0.$$

- $W = W(x,t) \in \mathbb{R}^m$: vector of conserved quantities;
- $x = (x^1 \dots x^d)$: space variable, d: space dimension, t: time;
- $\partial_t = \frac{\partial}{\partial t}$, $\partial_i = \frac{\partial}{\partial x_i}$;
- $F^i(W)$: flux vector (contains the physics). Hyperbolicity condition.
- Applications: Maxwell, compressible fluids, MHD, plasma physics, etc.

SCHNAPS: http://schnaps.gforge.inria.fr

- Factorize software developments: design of a generic, non-linear conservation laws solver.
- Optimizations for addressing hybrid CPU/GPU clusters.
- Fundamental and industrial applications.

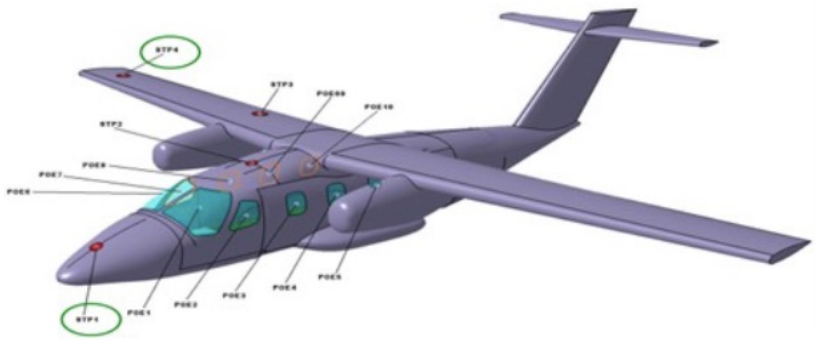
SCHNAPS: "Solveur Conservatif Hyperbolique Non-linéaire Appliqué aux PlaSmas".

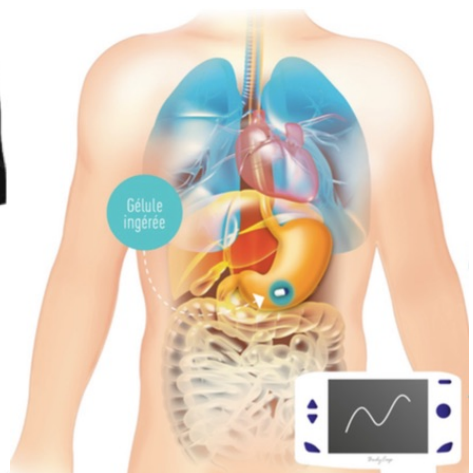
- C99, git, cmake, ctest, doxygen, GPL license.
- MPI for dealing with MIMD coarse grain parallelism.
- OpenCL: SIMD fine grain parallelism (multicore CPU or GPU).
- Task graph programming model.
- "Extreme programing" philosophy [Beck, 2000]: test driven development; short development cycles; pair programing; be prepared for changes; light documentation.

OpenCL specificities

- Similar to CUDA: handmade cache management (but more user-friendly cache systems are coming...); branching may be costly (SIMD parallelism).
- Industry standard: the very same program can really run on different accelerators. Drivers exist for: NVIDIA GPUs, AMD CPUs and GPUs, Intel CPUs and GPUs, MIC, ARM (CPU+GPU), IBM, etc.
- Kernel compilation at runtime: very interesting for metaprogramming and performance portability.
- Efficient OpenCL kernels are complex to design...

Unstructured grids



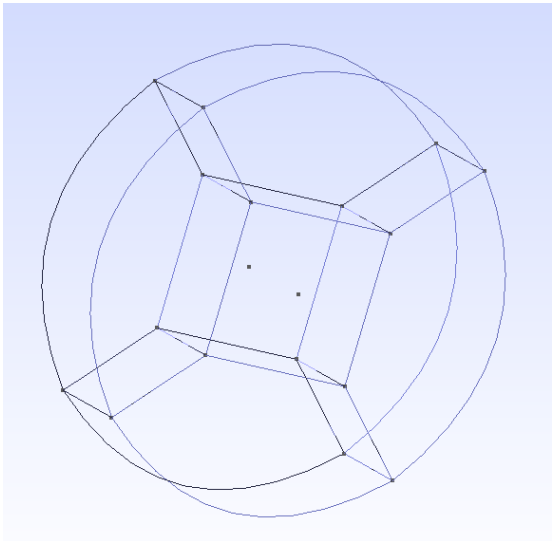


(Project with Thales, AxesSim, Body Cap, Citizens Sciences)

- Unstructured hexahedra mesh for representing complex geometries.
- Subdomain decomposition. 1 domain = 1 MPI node = OpenCL devices + CPU.
- Non-conformity is necessary.

Macromesh approach

- Geometry described by a coarse mesh made of hexahedral curved macrocells.



- The macromesh is known by all MPI nodes.

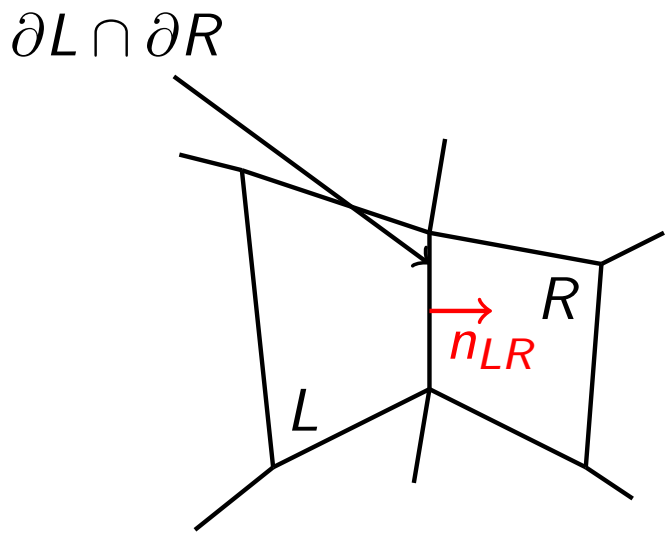
- Each macrocell is itself split into smaller subcells of size h. The subcell connectivity is not stored.
- In each subcell L we consider polynomial basis functions ψ_i^L of degree p.
- $W_L(x,t) \simeq \sum_i w_L^i(t)\psi_i^L(x)$.
- Possible non-conformity in "h" and "p".

Discontinuous Galerkin (DG) formulation

The numerical solution satisfies the DG approximation scheme

$$\forall L, \forall i \quad \int_L \partial_t W_L \psi_i^L - \int_L F(W_L, W_L, \nabla \psi_i^L) + \int_{\partial L} F(W_L, W_R, n_{LR}) \psi_i^L = 0.$$

- R denotes the neighbor cells along ∂L.
- n_{LR} is the unit normal vector on ∂L oriented from L to R.
- $F(W_L, W_R, n)$: numerical flux.
- $F(W, W, n) = \sum_k F^k(W) n_k$.

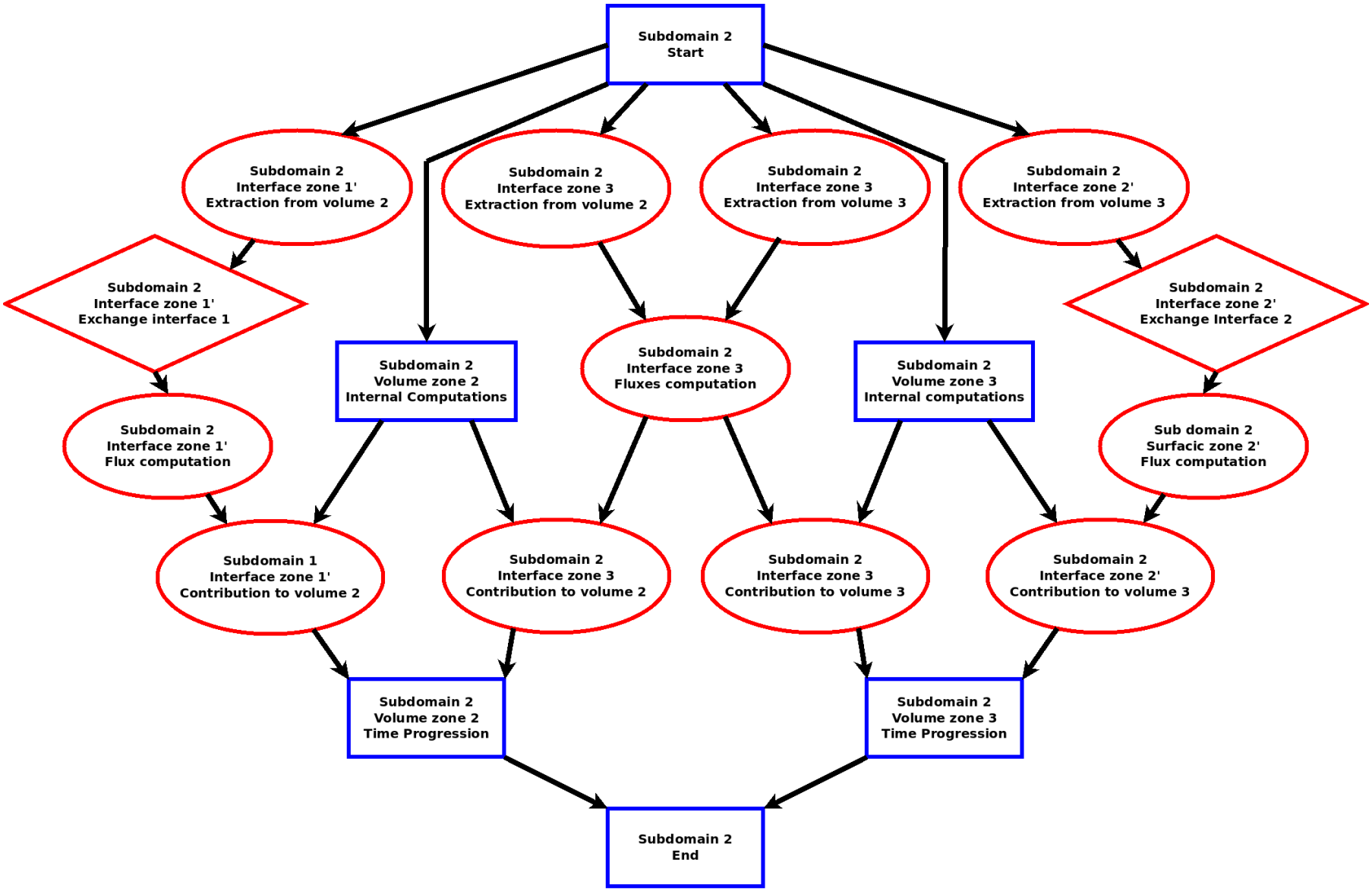


Explicit time integration of a system of ordinary differential equations.

Tasks

- Elementary tasks attached to macrocells or interfaces.
- A task is associated to a computational OpenCL kernel or to memory operations (GPU$\leftrightarrow$CPU and MPI transfers).
- A task graph for describing dependencies.
- The task graph is deduced only from the macromesh connectivity.
- Hand made task graph + OpenCL: complicated programming...

Task graph


Subdomain 2
Start
Subdomain 2
Interface zone 1'
Extraction from volume 2
Subdomain 2
Interface zone 3
Extraction from volume 2
Subdomain 2
Interface zone 3
Extraction from volume 3
Subdomain 2
Interface zone 2'
Extraction from volume 3
Subdomain 2
Interface zone 1'
Exchange interface 1
Subdomain 2
Interface zone 2'
Exchange Interface 2
Subdomain 2
Volume zone 2
Internal Computations
Subdomain 2
Interface zone 3
Fluxes computation
Subdomain 2
Volume zone 3
Internal computations
Subdomain 2
Interface zone 1'
Flux computation
Sub domain 2
Surfacic zone 2'
Flux computation
Subdomain 1
Interface zone 1'
Contribution to volume 2
Subdomain 2
Interface zone 3
Contribution to volume 2
Subdomain 2
Interface zone 3
Contribution to volume 3
Subdomain 2
Interface zone 2'
Contribution to volume 3
Subdomain 2
Volume zone 2
Time Progression
Subdomain 2
Volume zone 3
Time Progression
Subdomain 2
End

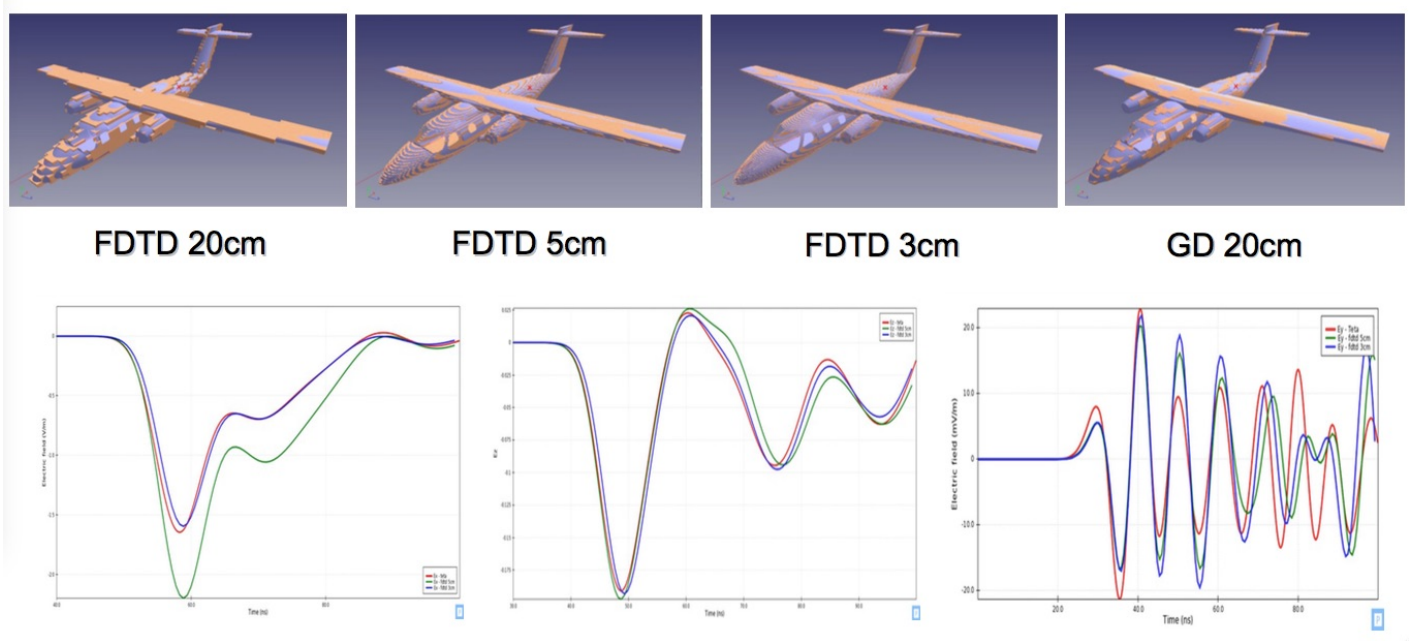
Sync./Async. comparison

Big mesh, polynomial order $D = 3$, NVIDIA K20 GPUs, infiniband network, single-precision floats.

		1 GPU	2 GPUs	4 GPUs	8 GPUs
Sync.	TFLOPS/s	1.01	1.84	3.53	5.07
ASync.	TFLOPS/s	1.01	1.94	3.74	7.26

Electromagnetic compatibility application [Cabel et al., 2011]

- Electromagnetic wave interaction with an aircraft.
- Aircraft geometry described with up to 3.5M hexaedrons ($\simeq$1 billion unknowns per time step): mesh of the interior and exterior of the aircraft. PML transparent boundary conditions.
- We use 8 GPUs to perform the computation. The biggest simulation does not fit into a single GPU memory.



1) Kinetic Schemes

Why is it important to solve transport equations ?

Vlasov-BGK framework

- Distribution function: $f(x, v, t)$, $x \in \mathbb{R}^d$, $v \in \mathbb{R}^d$, $t \in [0, T]$.
- Microscopic "collision vector" $K(v) \in \mathbb{R}^m$. Macroscopic conserved data

$$W(x, t) = \int_v f(x, v, t) K(v) dv.$$

- Entropy $s(f)$ and associated Maxwellian $m_W(v)$:

$$\int_v m_W K = W, \quad \int_v s(m_W) = \max_{\int_v fK = W} \left\{ \int_v s(f) \right\}.$$

- Vlasov-BGK equation ($a = a(x, t)$ is the acceleration):

$$\partial_t f + v \cdot \nabla_x f + a \cdot \nabla_v f = \eta (m_W - f).$$

Kinetic schemes

When the relaxation parameter η is big, the Vlasov equation provides an approximation of the hyperbolic conservation laws

$$\partial_t W + \nabla \cdot F(W) + S(W) = 0,$$

with

$$F^i(W) = \int_v v^i m_W(v) K(v) dv.$$

$$S(W) = a \cdot \int_v \nabla_v m_W(v) K(v) = -a \cdot \int_v m_W(v) \nabla_v K(v).$$

Main idea: numerical solvers for the linear scalar transport equation lead to natural solvers for the non-linear hyberbolic system [Deshpande, 1986].
Many applications: fluid mechanics, sprays, plasmas, MHD and even Maxwell !

Example I: compressible fluids

The Maxwellian m_W has not necessarily a physical meaning.
Famous example [Perthame, 1990]

$$W = \begin{pmatrix} \rho \\ \rho u \\ \rho e + \rho u^2/2 \end{pmatrix}, \quad F(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \left(\rho e + \rho u^2/2 + p\right) u \end{pmatrix}, \quad p = 2\rho$$

It is possible to find a convex entropy and a kinetic interpretation with

$$a = 0, \quad K(v) = \begin{pmatrix} 1 \\ v \\ v^2/2 \end{pmatrix}, \quad m_W(v) = \frac{\rho}{2\sqrt{6e}} \chi_{[-1,1]}\left(\frac{v-u}{\sqrt{6e}}\right),$$

where $\chi_{[-1,1]}$ is the indicator function of $[-1,1]$.

Example II: Lattice Boltzmann

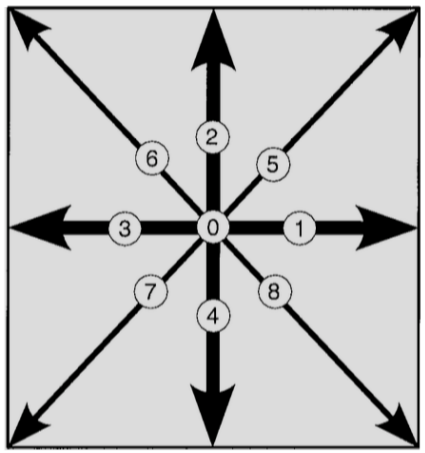
We would like to solve a transport equation for each v. How to reduce as much as possible the velocity space ?
Answer: lattice Boltzmann. Example for a barotropic low-Mach inviscid fluid.

- Density ρ
- velocity $u = (u^1, u^2)$
- pressure $p = c^2\rho$ (c is the sound speed)

$$\partial_t \rho + \nabla \cdot (\rho u) = 0,$$
$$\partial_t (\rho u) + \nabla \cdot (p I + \rho u \otimes u) = 0.$$

Lattice kinetic interpretation [Qian et al., 1992]

Lattice kinetic interpretation under a low Mach hypothesis: $|u| \ll c$



- In 2D, $N = 9$, $v \in \{\xi_0 \dots \xi_{N-1}\}$. In 3D, $N = 27$.
- $\int_v = \sum_{i=0}^{N-1}$
- $W = (\rho, \rho u)$,$K(\xi) = (1, \xi)$, $a = 0$,

$$m_W(\xi_i) = \rho\omega_i \left(1 + \frac{\xi_i \cdot u}{\theta} + \frac{(\xi_i \cdot u)^2}{\theta} - \frac{u^2}{2\theta}\right),$$

$$\theta = 1/3, \quad \omega_0 = 4/9, \quad \omega_{1-4} = 1/9, \quad \omega_{5-8} = 1/36.$$

Implicit DG solver for transport

Explicit Discontinuous Galerkin (DG) solvers are constrained by an annoying CFL condition. Empirical stability condition

$$\Delta t \leq \frac{\Delta x}{2d(2p+1)V_{\max}}$$

with:

- Δx: cell size.
- d: space dimension
- p : polynomial degree
- $V_{\max}$: maximal speed
- Can be worse...

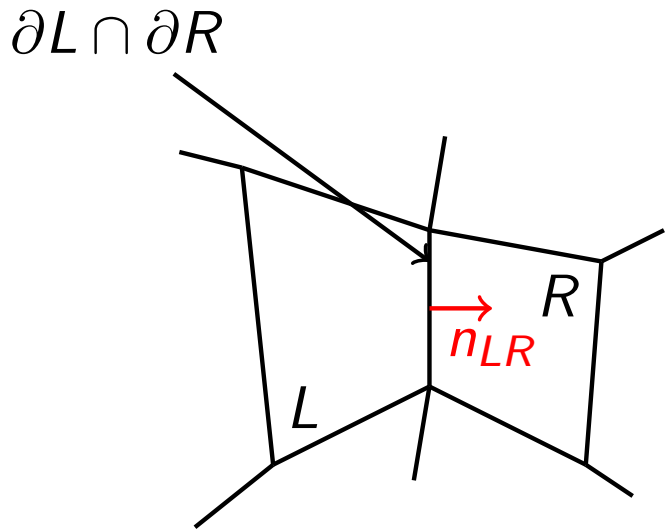
Implicit solvers have no time-step restriction.

Implicit DG for transport equation

Implicit DG approximation scheme: $\forall L, \forall i$

$$\int_L \frac{f_L^n - f_L^{n-1}}{\Delta t} \psi_i^L - \int_L v \cdot \nabla \psi_i^L f_L^n + \int_{\partial L} \left(v \cdot n^+ f_L^n + v \cdot n^- f_R^n \right) \psi_i^L = 0.$$

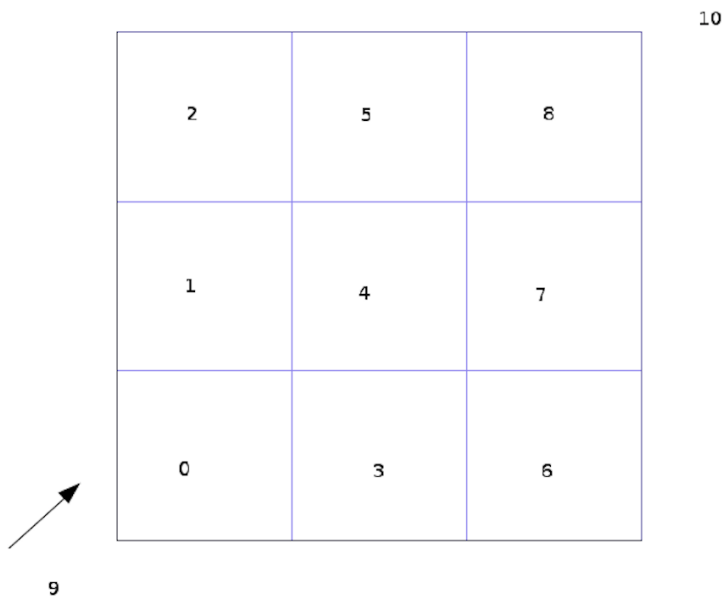
- R denotes the neighbor cells along ∂L.
- $v \cdot n^+ = \max(v \cdot n, 0)$, $v \cdot n^- = \min(v \cdot n, 0)$.
- n_{LR} is the unit normal vector on ∂L oriented from L to R.



Second order: Crank-Nicolson, improved Euler, etc.

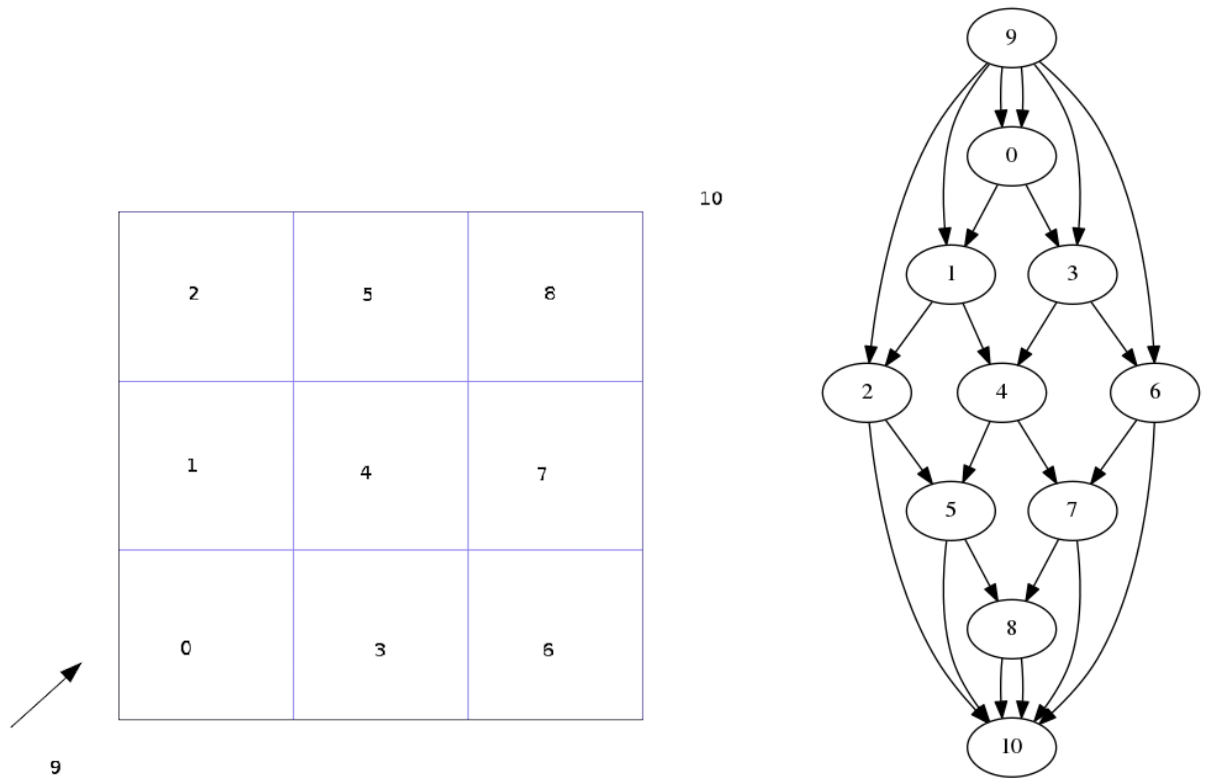
Upwind numbering

- L is *upwind* with respect to R if $v \cdot n_{LR} > 0$ on $\partial L \cap \partial R$.
- In a macrocell L, the solution depends only on the values of f in the upwind macrocells.
- No assembly and factorization of the global system.



Dependency graph

For a given velocity v we can build a dependency graph. Vertices are associated to macrocells and edges to macrocells interfaces or boundaries. We consider two fictitious additional vertices: the "upwind" vertex and the "downwind" vertex.



Upwind implicit algorithm

[Duff and Reid, 1978, Johnson et al., 1984, Wang and Xu, 1999, Natvig and Lie, 2008]

- Topological ordering of the dependency graph (supposed to be a Direct Acyclic Graph).
- First time step: Assembly and LU decomposition of the local macrocell matrices.
- For each macrocell (in topological order):
  - Compute volume terms.
  - Compute upwind fluxes.
  - Solve the local linear system.
  - Extract the results to the downwind cells.

Parallelization ?

3) StarPU parallelization

How to handle in practice a non-uniform parallelism ?

StarPU

- StarPU is a library developed at Inria Bordeaux [Augonnet et al., 2012]: `http://starpu.gforge.inria.fr`
- Data-based task parallelism.
- Task description: codelets, input data (R), output data (W or RW).
- The task graph is automatically inferred from data dependency.
- The user submits tasks in a correct sequential order.
- StarPU schedules the tasks in parallel if possible.

StarPU implementation

- We start from a working sequential code.
- StarPU implementation was smooth: incremental migrations task by task.
- Several implementations of the same task are possible (CPU, optimized CPU, GPU OpenCL, GPU CUDA, MIC, etc.)

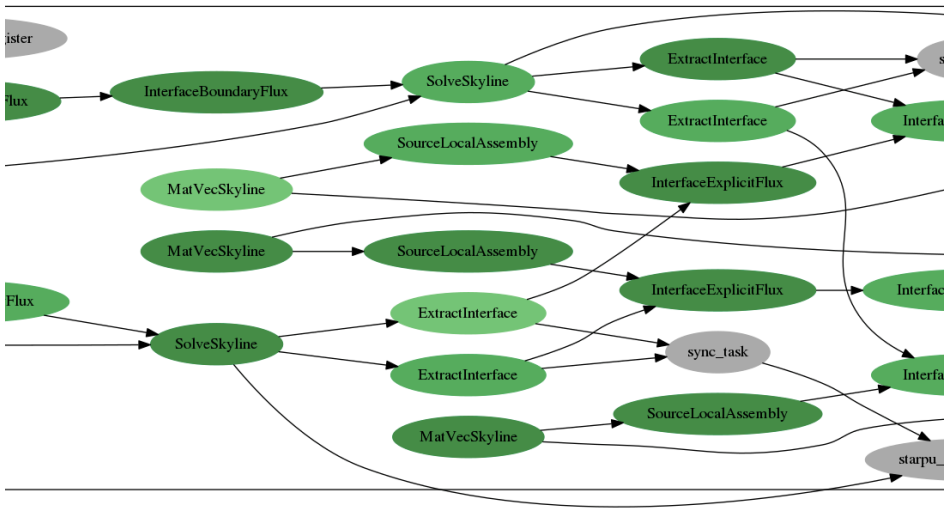
Preliminary results

We compare a global direct solver to the upwind StarPU solver with several meshes.
Weak scaling. "dmda" scheduler. AMD Opteron 16 cores, 2.8 Ghz.
Timing in seconds for 200 iterations.

nb cores	0	1	2	4	8	16
$10 \times 10 \times 8 \times 8$ direct	30	144	-	-	-	-
$10 \times 10 \times 8 \times 8$ upwind	-	32	19	12	7	6
$20 \times 20 \times 4 \times 4$ upwind	-	41	26	17	12	17
$20 \times 20 \times 8 \times 8$ upwind	-	120	72	40	28	20

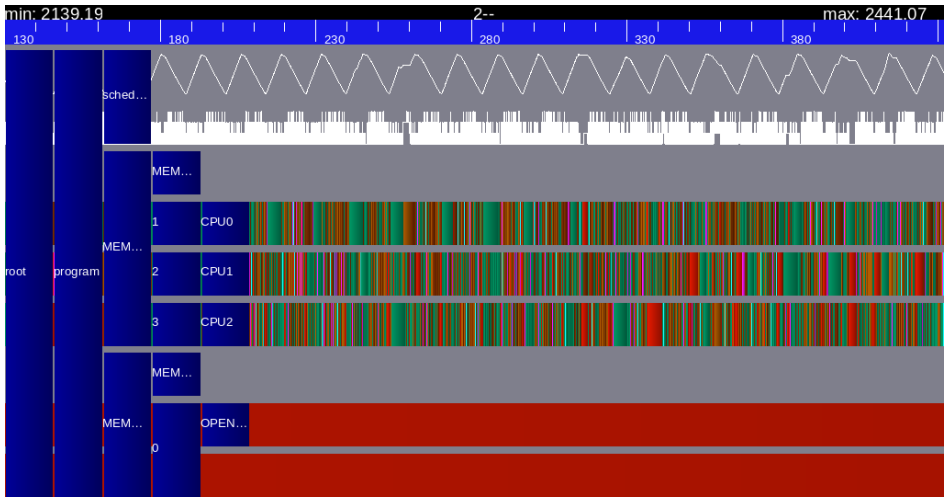
Task graph

Zoom of the task graph generated by StarPU



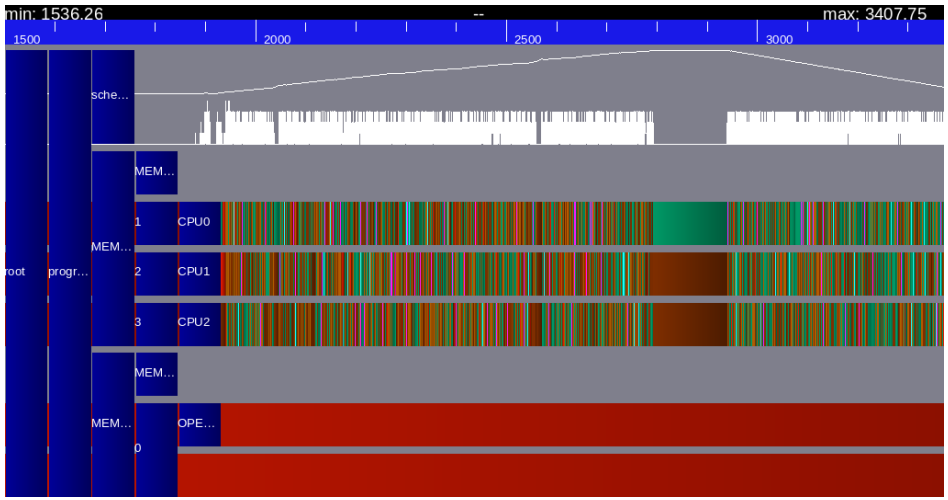
Gantt diagram

Gantt diagram generated by StarPU: sync point at the end of each time step



Gantt diagram

Gantt diagram generated by StarPU: without sync point at the end of each time step



Conclusion

My (current) philosophy of software design:

- Mathematics: use the same framework for several applications.
- Extreme programing: avoid software with a rigid universe.
- data-based task parallelism: move gently to hybrid parallelism.

Bibliography I

[Augonnet et al., 2012] Augonnet, C., Aumage, O., Furmento, N., Namyst, R., and Thibault, S. (2012).
StarPU-MPI: Task Programming over Clusters of Machines Enhanced with Accelerators.
In Jesper Larsson Träff, S. B. and Dongarra, J., editors, *EuroMPI 2012*, volume 7490 of *LNCS*. Springer.
Poster Session.

[Beck, 2000] Beck, K. (2000).
Extreme programming explained: embrace change.
Addison-Wesley Professional.

[Cabel et al., 2011] Cabel, T., Charles, J., and Lanteri, S. (2011).
Multi-GPU acceleration of a DGTD method for modeling human exposure to electromagnetic waves.

[Deshpande, 1986] Deshpande, S. (1986).
Kinetic theory based new upwind methods for inviscid compressible flows.
In *24th AIAA Aerospace Sciences Meeting*, volume 1.

[Duff and Reid, 1978] Duff, I. S. and Reid, J. K. (1978).
An implementation of tarjan's algorithm for the block triangularization of a matrix.
ACM Transactions on Mathematical Software (TOMS), 4(2):137–147.

[Johnson et al., 1984] Johnson, C., Nävert, U., and Pitkäranta, J. (1984).
Finite element methods for linear hyperbolic problems.
Computer methods in applied mechanics and engineering, 45(1):285–312.

[Natvig and Lie, 2008] Natvig, J. R. and Lie, K.-A. (2008).
Fast computation of multiphase flow in porous media by implicit discontinuous galerkin schemes with optimal ordering of elements.
Journal of Computational Physics, 227(24):10108–10124.

Bibliography II

[Perthame, 1990] Perthame, B. (1990).
Boltzmann type schemes for gas dynamics and the entropy property.
SIAM Journal on Numerical Analysis, 27(6):1405–1421.

[Qian et al., 1992] Qian, Y., d'Humières, D., and Lallemand, P. (1992).
Lattice bgk models for navier-stokes equation.
EPL (Europhysics Letters), 17(6):479.

[Wang and Xu, 1999] Wang, F. and Xu, J. (1999).
A crosswind block iterative method for convection-dominated problems.
SIAM Journal on Scientific Computing, 21(2):620–645.