Asynchronous OpenCL/MPI numerical simulations of conservation laws

Philippe HELLUY [1,3], Thomas STRUB [2].

[1]IRMA, Université de Strasbourg, [2]AxesSim, [3]Inria Tonus, France

IWOCL 2015, Stanford

Conservation laws

Many equations in physics are systems of conservation laws

$$\frac{\partial}{\partial t} W + \sum_{i=1}^{d} \frac{\partial}{\partial x^i} F^i(W) = 0.$$

- $W = W(x,t) \in \mathbb{R}^m$: vector of conserved quantities, $F^i(W)$: flux vector.
- $x = (x^1 \dots x^d)$: space variable, d: space dimension, t: time;

Applications: fluid mechanics, electromagnetics, ...

Outlines:

1. 2D structured grids, synchronous OpenCL/MPI based numerical simulations.
2. 3D unstructured grids, asynchronous OpenCL/MPI based numerical simulations.

1) Structured grid

First simple approach: discretization of a 2D equation ($d = 2$) on a structured grid

$$\frac{\partial}{\partial t} W + \frac{\partial}{\partial x^1} F^1(W) + \frac{\partial}{\partial x^2} F^2(W) = 0.$$

- Grid step: Δx
- We compute samples $W^n_{i,j}$ of $W(x,t)$ at grid points $x = (i\Delta x, j\Delta x)$ and time $t = n\Delta t$.
- Finite Difference (FD) method:

$$\frac{W^*_{i,j} - W^n_{i,j}}{\Delta t} + \frac{F^{1,n}_{i+1/2,j} - F^{1,n}_{i-1/2,j}}{\Delta x} = 0,$$

$$\frac{W^{n+1}_{i,j} - W^*_{i,j}}{\Delta t} + \frac{F^{2,n}_{i,j+1/2} - F^{2,n}_{i,j-1/2}}{\Delta x} = 0.$$

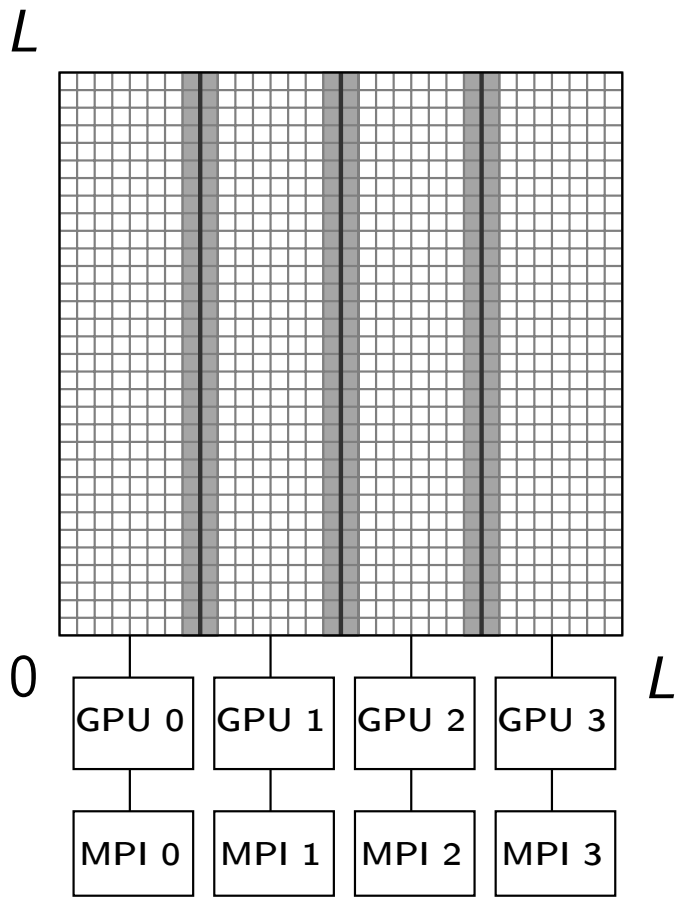
OpenCL implementation

The data are arranged in a (i, j) matrix. 1 work-item = 1 cell (i, j).
1 work-group = 1 row i.
For each time step n:

- compute the fluxes balance in the x^1-direction for each cell of each row of the grid.
- transpose the matrix (exchange i and j) in a coalescent way.
- compute the fluxes balance in the x^2-direction for each row of the transposed grid.
- transpose again the matrix.

OpenCL + synchronous MPI

- Use of several GPUs;
- Subdomain decomposition;
- 1 GPU = 1 subdomain = 1 MPI node;
- MPI for exchanging data between GPUs (greyed cells layers).



Comparisons

On large grids ($> 1024 \times 1024$). We compare:

- an optimized OpenMP implementation of the FD scheme on 2x6-core CPUs;
- the OpenCL implementation running on 2x6-core CPUs, NVidia or AMD GPU;
- the OpenCL+MPI implementation running on 4 GPUs.

Implementation	Time	Speedup
OpenMP (CPU Intel 2x6 cores)	717 s	1
OpenCL (CPU Intel 2x6 cores)	996 s	0.7
OpenCL (NVidia Tesla K20)	45 s	16
OpenCL (AMD Radeon HD 7970)	38 s	19
OpenCL + MPI (4 x NVIDIA K20)	12 s	58

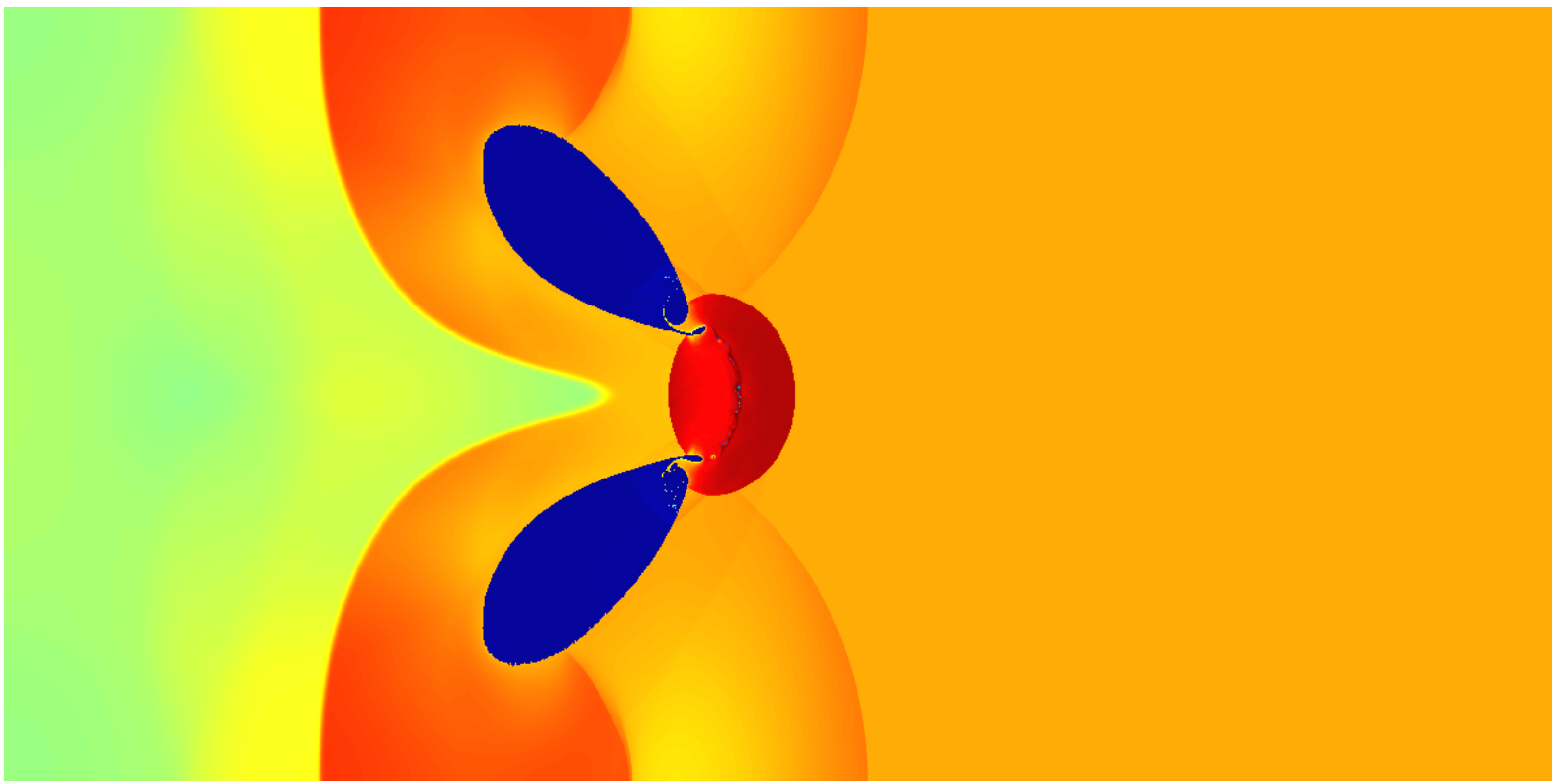
Shock-bubble interaction

- Simulation of a compressible two-fluid flow: interaction of a shock wave in a liquid with a gas bubble
- Coarse mesh OpenCL simulation on an AMD HD 5850
- OpenGL/OpenCL interop + video capture.

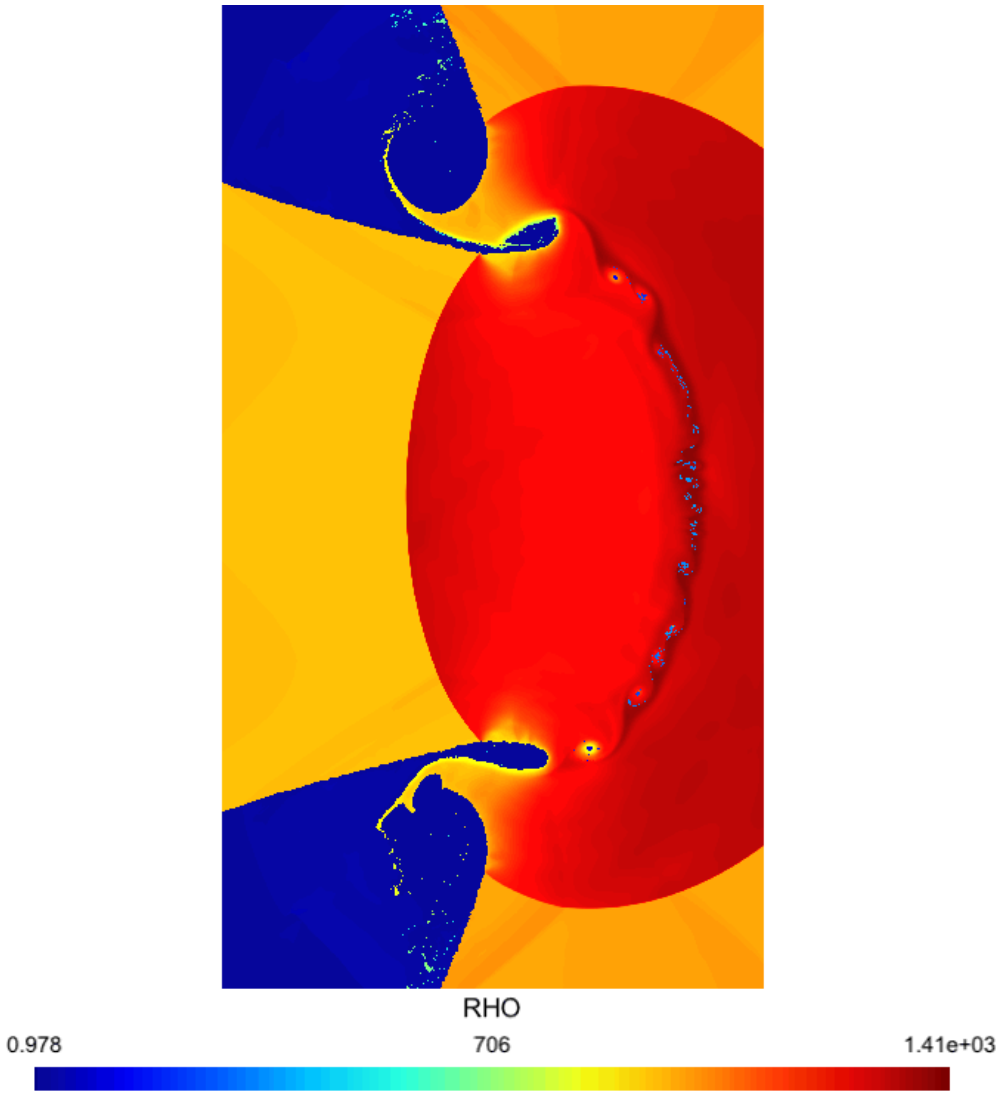
`https://www.youtube.com/watch?v=c8hcqihJzbw`

Very fine mesh

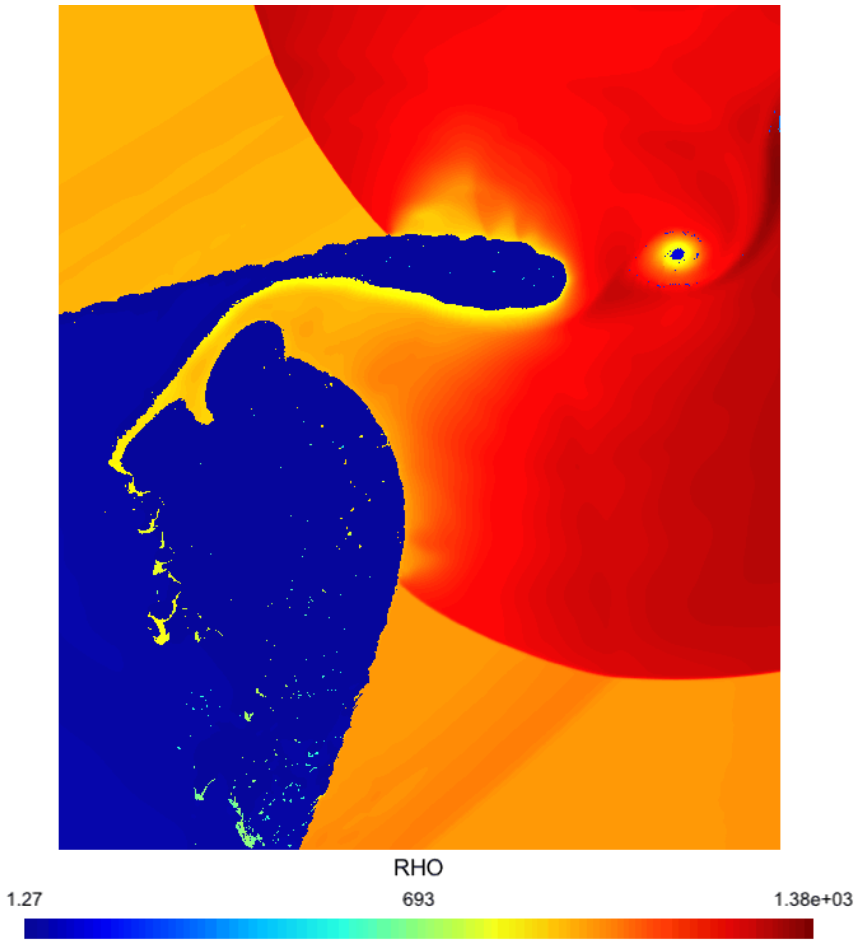
- Very fine mesh OpenCL + MPI simulation, 40,000x20,000 grid. 4 billions unknowns per time step
- 10xNVIDIA K20 GPUs, 30 hours
- Red=high density (compressed liquid); blue=low density (gas).

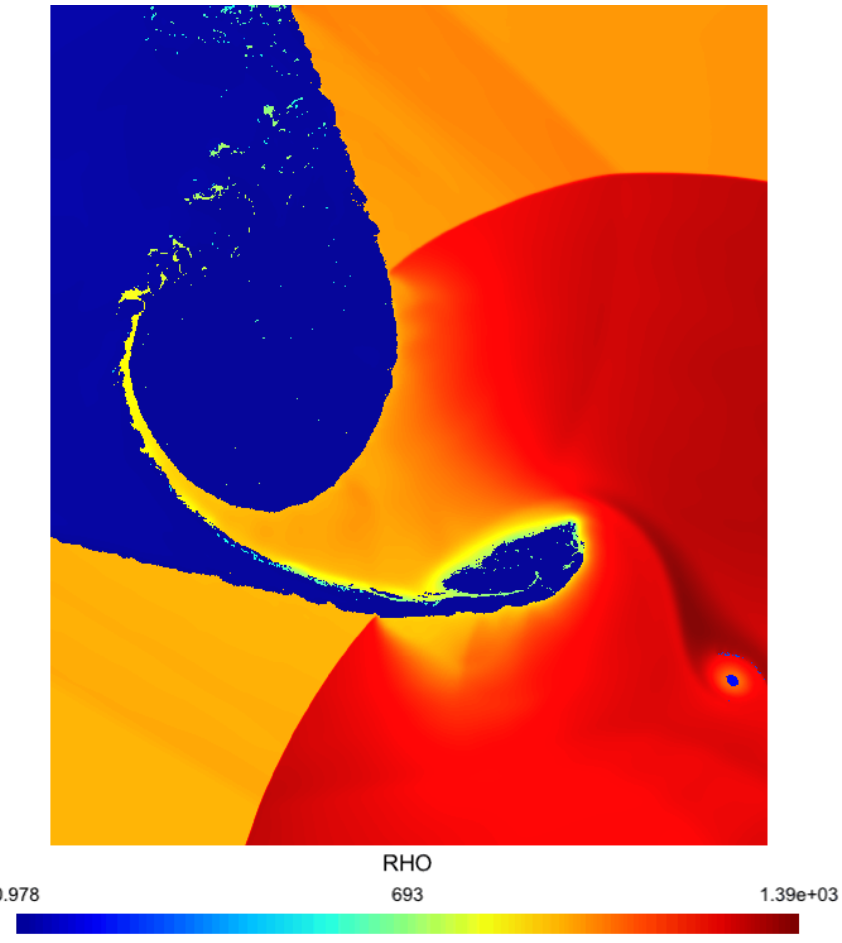


Zoom 1



Zoom 2





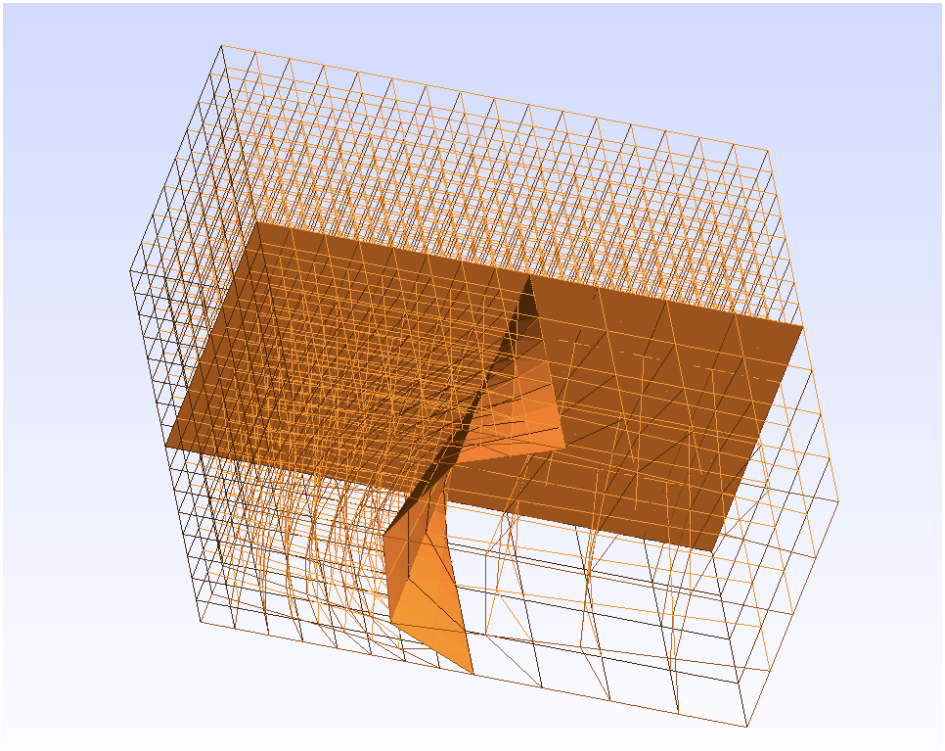
2) Unstructured grid

- Unstructured hexahedrons mesh for representing complex geometries.
- Subdomain decomposition. 1 domain = 1 MPI node = 1 OpenCL device.
- Zone decomposition. Each subdomain is split into volume zones and interface zones.
- Non-conformity between zones is allowed.

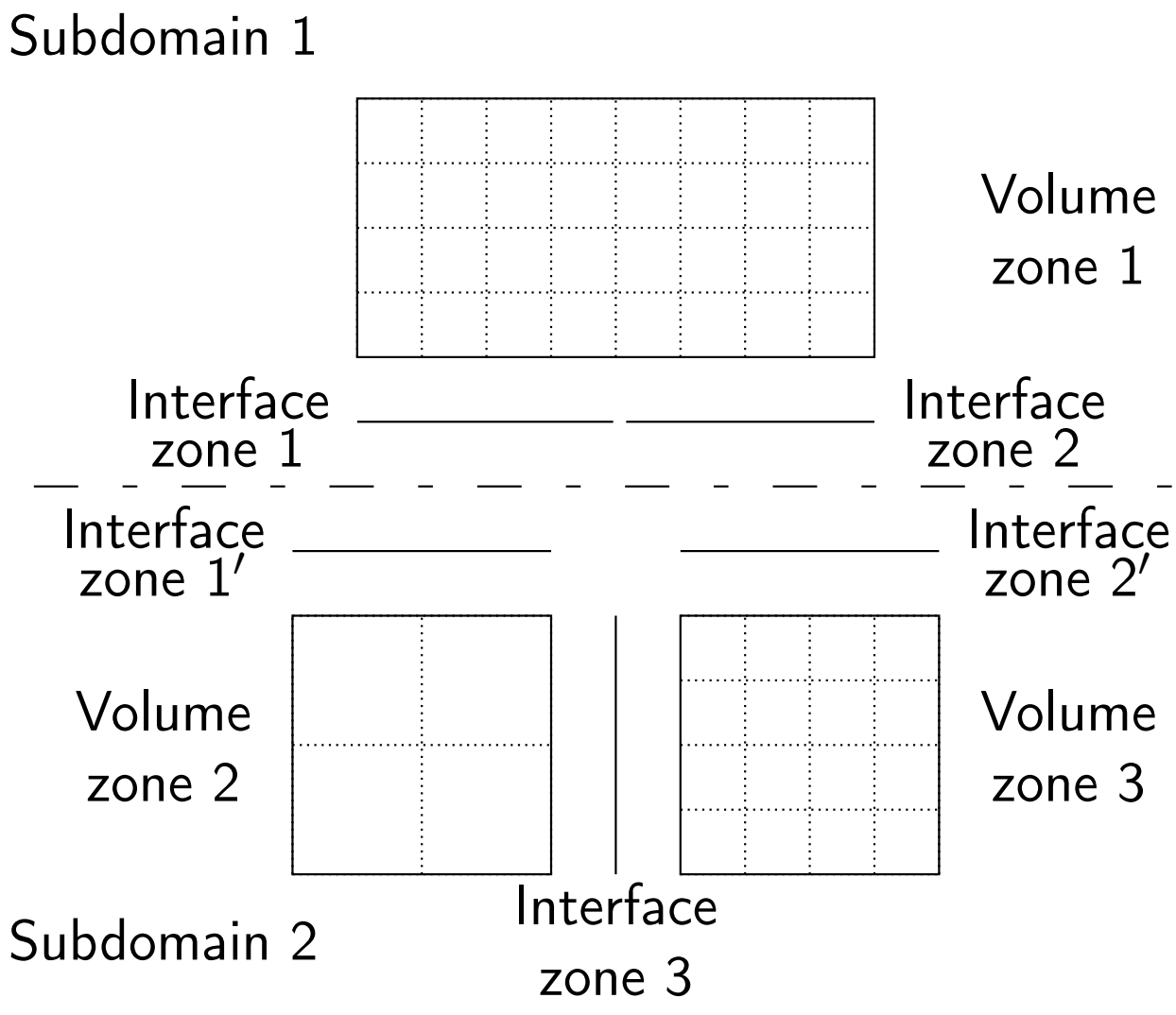
Mesh example

A non-conforming mesh with two subdomains, three volume zones and three interface zones.

- Subdomain 1: only one big refined volume zone. Two interface zones.
- Subdomain 2: two small volume zones (coarse and refined). Three interface zones.



Mesh structure

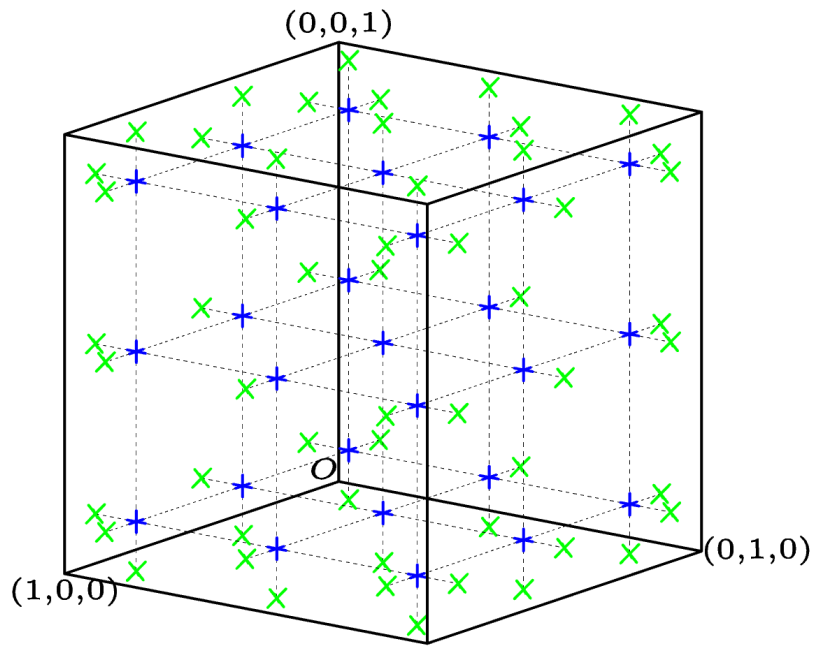


Discontinuous Galerkin (DG) approximation

Generalization of the FD method: DG method in a 3D space.
In each cell L of the mesh, the conserved quantities are expanded on Lagrange polynomial basis functions

$$W(x,t) = W_L^j(t)\psi_j^L(x), \quad x \in L.$$

- L is a (possibly stretched) hexahedron
- W is determined by its values at the blue points
- W is discontinuous at green points.

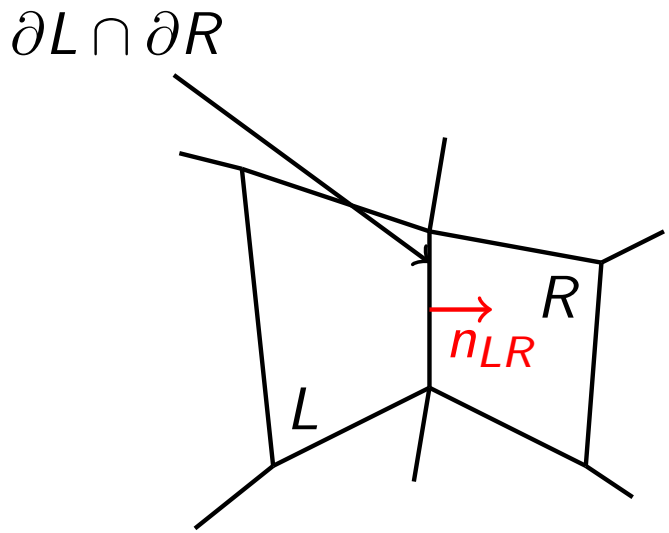


DG formulation

The numerical solution satisfies the DG approximation scheme

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

- 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.
- $F(W, W, n) = F^k(W) n_k$.



Time integration of a system of ordinary differential equations.

OpenCL + asynchronous MPI

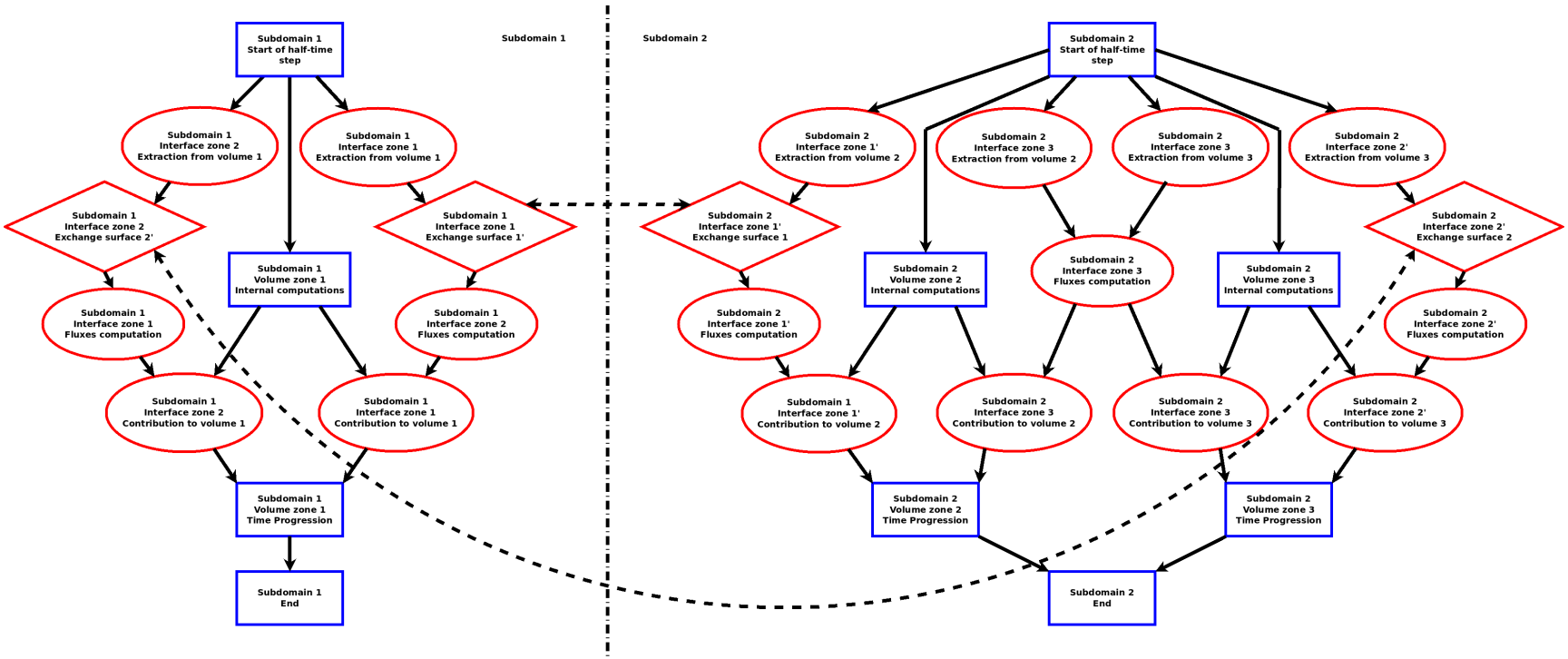
advantages of the DG formulation:

- new possibilities: varying polynomial order, mesh refinement, complex geometries.
- local stencil.
- high polynomial order $\Rightarrow$ high amount of uniform local parallel computations.
- many optimizations for hexahedrons meshes.
- natural MIMD/SIMD parallelism: subdomains (MPI), elementary computations (OpenCL).

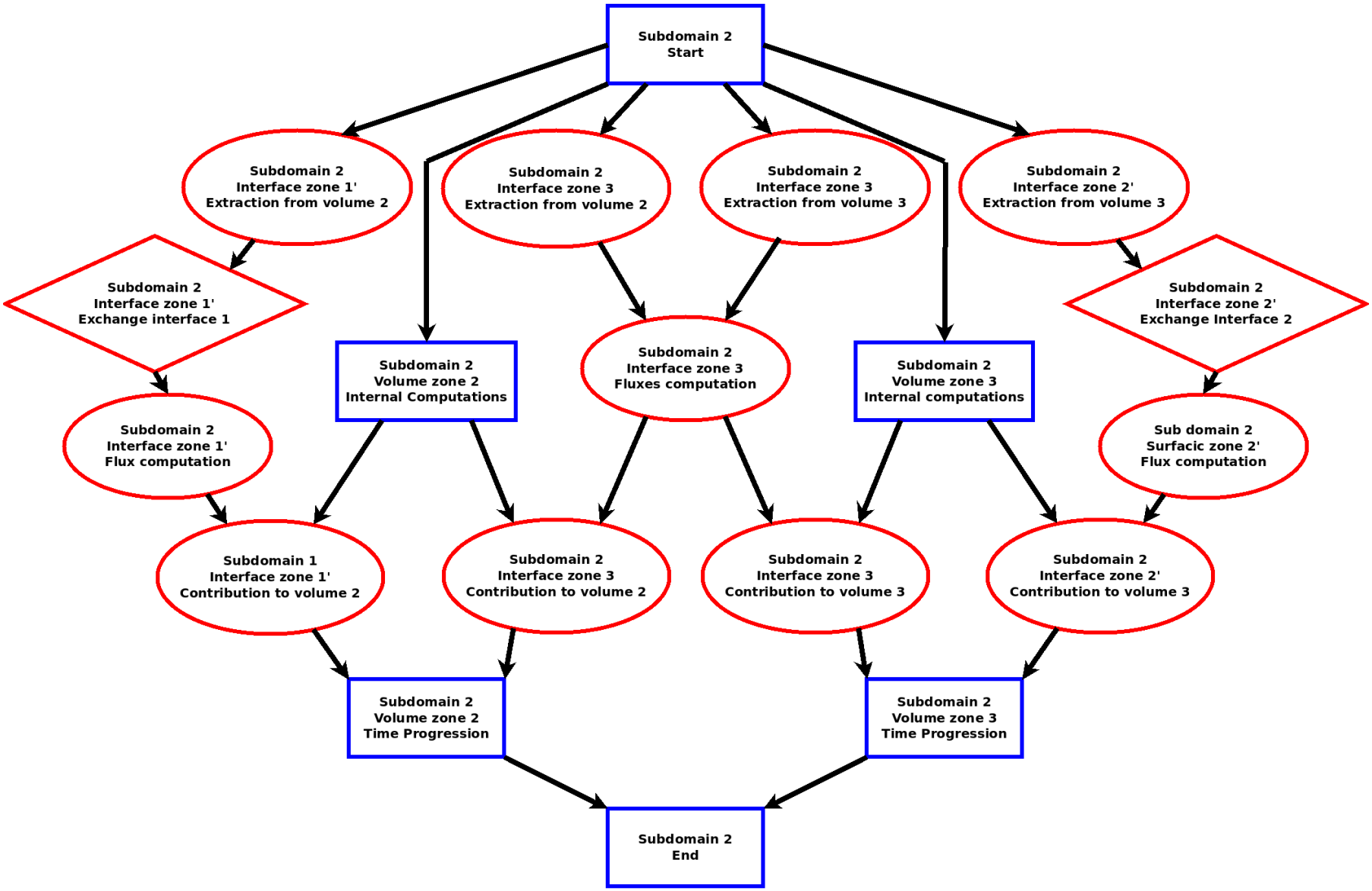
possible issues:

- memory access (unstructured mesh) at interfaces between cells $\rightarrow$ hard to avoid...
- branch tests in GPU kernels $\rightarrow$ compile a customized kernel for each zone.
- MPI communications imply GPU/Host memory transfers $\rightarrow$ overlap transfers and computations.

Task graph


Subdomain 1
Subdomain 2
Subdomain 1 Start of half-time step
Subdomain 1 Interface zone 2 Extraction from volume 1
Subdomain 1 Interface zone 1 Extraction from volume 1
Subdomain 1 Interface zone 2 Exchange surface 2'
Subdomain 1 Interface zone 1 Exchange surface 1'
Subdomain 1 Volume zone 1 Internal computations
Subdomain 1 Interface zone 1 Fluxes computation
Subdomain 1 Interface zone 2 Fluxes computation
Subdomain 1 Interface zone 2 Contribution to volume 1
Subdomain 1 Interface zone 1 Contribution to volume 1
Subdomain 1 Volume zone 1 Time Progression
Subdomain 1 End
Subdomain 2 Start of half-time step
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 surface 1
Subdomain 2 Interface zone 2' Exchange surface 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' Fluxes computation
Subdomain 2 Interface zone 2' Fluxes 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

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

MPI/OpenCL events management

Problem: how to express the dependency between MPI and OpenCL operations ?

- We decided to rely only on the OpenCL events management.
- The beginning of a task depends on the completions of a list of OpenCL events. The task is itself associated to an OpenCL event.
- At an interface zone between two subdomains, an extraction task contains a GPU to host memory transfer, a MPI send/receive communication and a host to GPU transfer.
- we create an OpenCL user event, and launch a MPI blocking sendrecv in a thread. At the end of the communication, in the thread, the OpenCL event is marked as completed. Using threads avoids blocking the main program flow.

Results

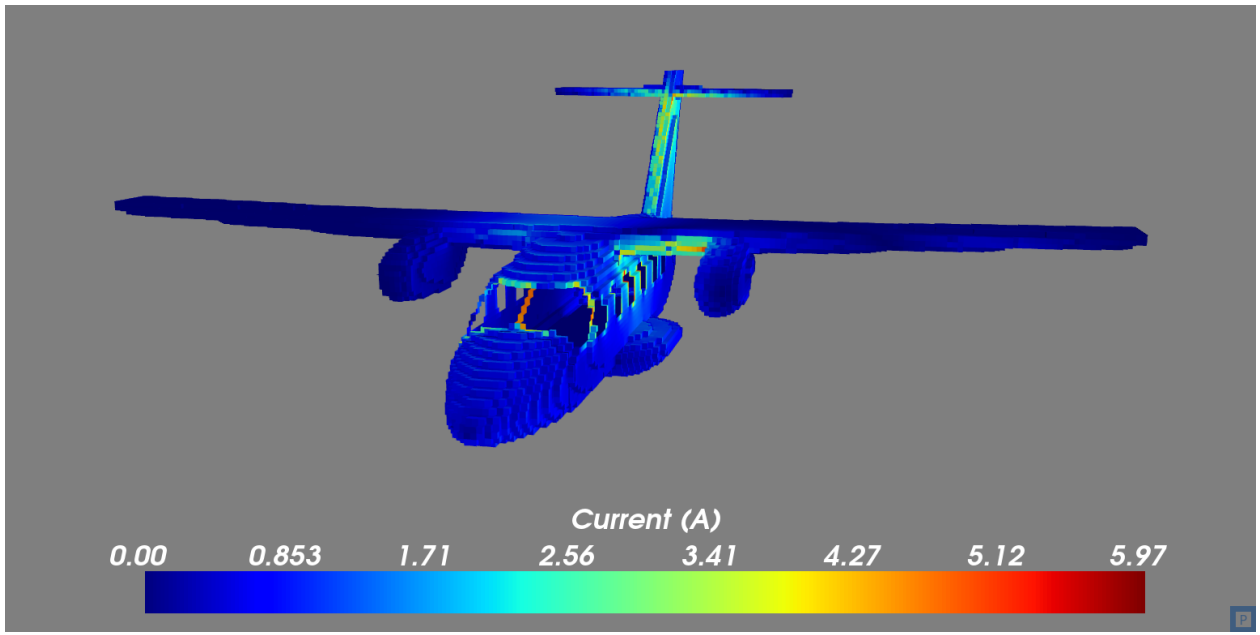
Big mesh, polynomial order $D = 3$, NVIDIA K20 GPUs, infiniband network.

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

We achieve $\simeq 30\%$ of the peak performance.

Application

- Electromagnetic wave interaction with an aircraft (Maxwell equations).
- Aircraft geometry described with $3,337,875$ hexaedrons ($\simeq$1 billion unknowns per time step): mesh of the interior and exterior of the aircraft.
- We use 8 GPUs to perform the computation. The simulation does not fit into a single GPU memory.



Conclusion

- Many physical models are conservation laws. Among them: two-fluid flows, electromagnetics.
- Efficient OpenCL/MPI computing requires adapted data structures.
- OpenCL allows driving asynchronous computations and MPI communications.
- Work in progress: optimizing unstructured memory access, more sophisticated runtime, new physical models.

For more details and a bibliography see:
`https://hal.archives-ouvertes.fr/hal-01134222v2`