

Workshop Cargèse

Trends in physical and numerical modeling of multiphase flows,
CFD and its experimental validation for multiphase flows

IESC Cargèse 2012, sep 24-28 2012

Some examples for CFD instant computations on GPU
LBM for incompressible fluids
VFM for compressible fluids

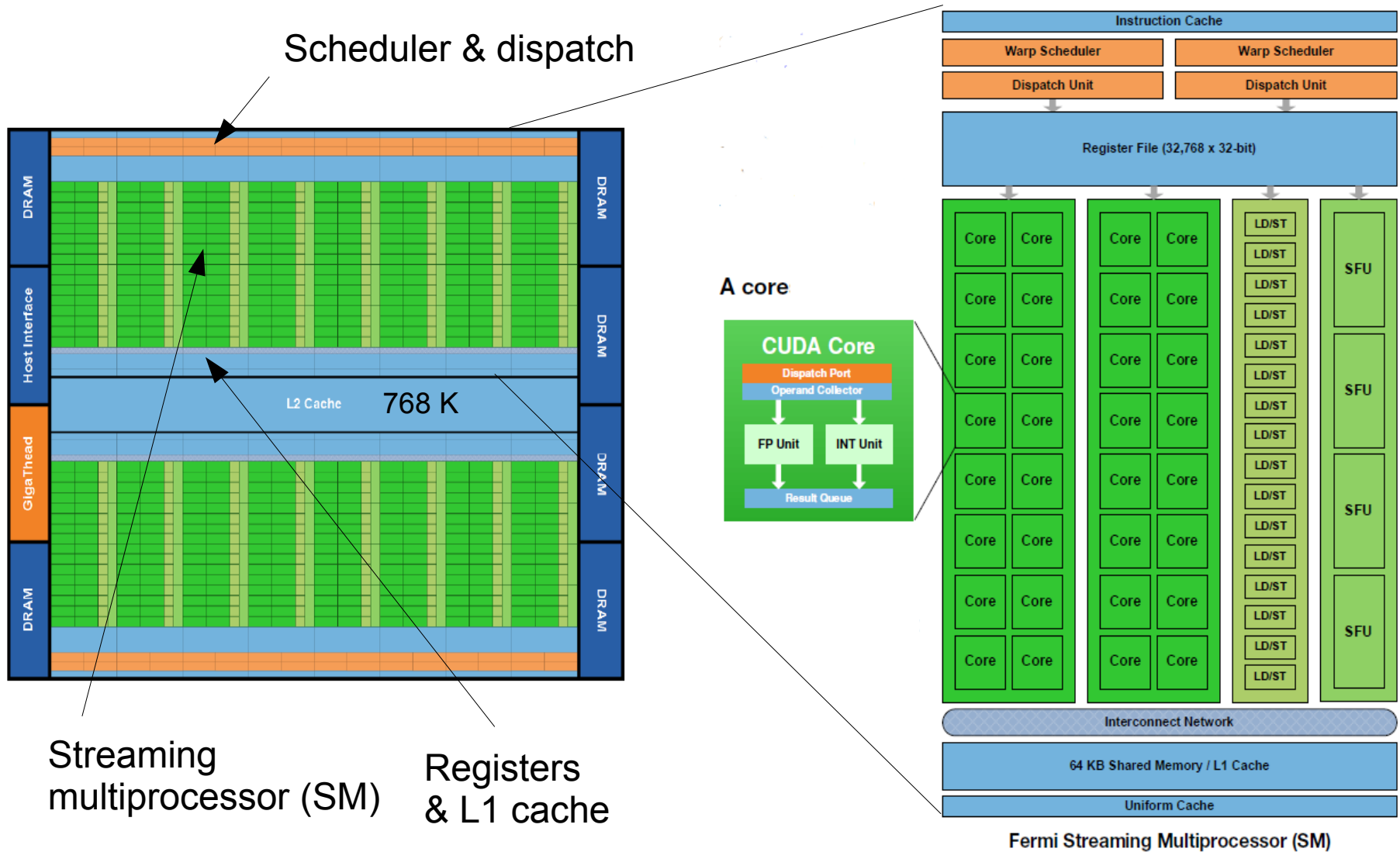
Florian De Vuyst, Christophe Labourdette

Centre de Mathématiques et de leurs Applications CMLA,
CNRS UMR 8536, ENS Cachan

Outline

1. Lattice Boltzmann method for 2D unsteady Navier-Stokes equations
2. Finite Volume method for 2D compressible Euler equations
3. Experience feedback
4. Work in progress
5. Demo !

nVIDIA GPU *FERMI* compute architecture



Hardware in use at CMLA

- PC Workstation with nVIDIA TESLA C2070 (nVIDIA Equipment Grant 2011)



448 cores, FERMI
6 GB Mem
250W

512 GFlops Double Precision (peak)
1,03 TFlops Single Precision (peak)

- Gamer PC Dell Alienware, nVIDIA GTX 580M



384 cores, FERMI
2.0 GB DDR5 Mem

950 GFlops Single Precision (peak)

Brand new nVIDIA GPU (summer 2012) ...

NVIDIA GeForce GTX 690

- 3072 cores !!
- Base Clock 915 MHz
- Boost Clock 1019 Mhz
- Memory bandwidth 6.0 Gbps
- Mem 4096 MB (2048 MB per GPU)
- DRAM bus memory 512-bit GDDR5



Theoretical peak performance :
about 6 TFlops !! (single precision)

1. Lattice Boltzmann method for 2D unsteady Navier-Stokes equations

- A mesoscopic paradigm to model PDE problems
- A « smoothed » Lattice Gas automaton model
- Navier-Stokes case : based on a very simple approximation and discretization of the Boltzmann-type equation

Lattice BGK model

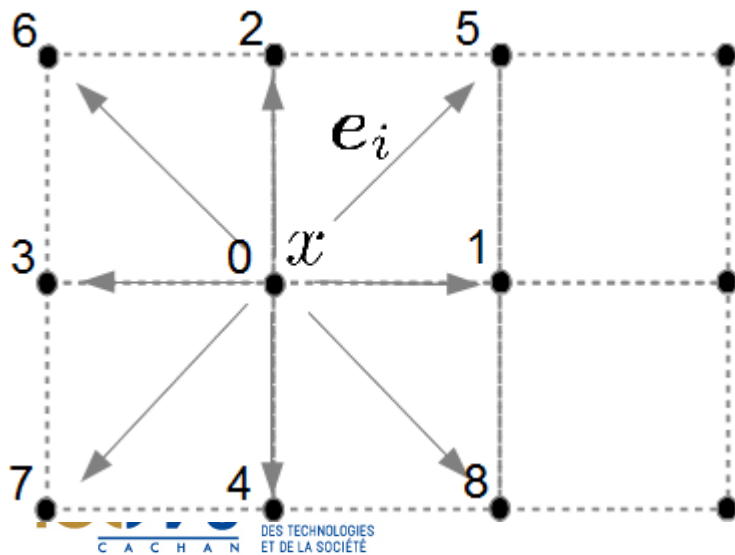
Based on a simple discretization of the Boltzmann equation with BGK approximation for the collision term :

$$f(x + \mathbf{e}_i \Delta t, \mathbf{e}_i, t + \Delta t) - f(x, \mathbf{e}_i, t) = \frac{1}{\tau} [f_i^{eq}(\rho, \mathbf{u}) - f(x, \mathbf{e}_i, t)]$$

$$\Delta x = \Delta t = 1$$

Relaxation toward some equilibrium...

$$\mathbf{e}_0 = 0, \mathbf{e}_1 = (1, 0), \mathbf{e}_2 = (1, 1), \dots$$



Moments:

$$\sum_{i \in \mathcal{S}} f(x, \mathbf{e}_i, t) = \rho(x, t),$$

$$\sum_{i \in \mathcal{S}} \mathbf{e}_i f(x, \mathbf{e}_i, t) = \rho \mathbf{u}(x, t).$$

Lattice BGK model

- Unknowns $f_i(x, t) \equiv f(x, \mathbf{e}_i, t)$, $i \in \{0, \dots, 8\}$.

- LB equation

$$f_i(x + \mathbf{e}_i, t + \Delta t) = f_i(x, t) + \omega [f_i^{eq}(\rho(x, t), \mathbf{u}(x, t)) - f_i(x, t)]$$

- Requirements

$$\omega = \frac{1}{\tau}.$$

$$\sum_{i \in \mathcal{S}} f_i^{eq}(\rho, \mathbf{u}) = \rho,$$

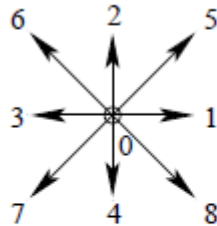
+ some galilean invariance conditions

$$\sum \mathbf{e}_i f_i^{eq}(\rho, \mathbf{u}) = \rho \mathbf{u}.$$

2D and 3D lattices & equilibrium distribution

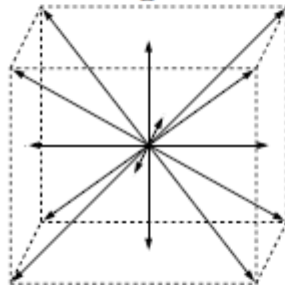
D2Q9
$$f_i^{eq} = \rho w_i \left[1 + 3\mathbf{u} \cdot \mathbf{e}_i + \frac{9}{2}(\mathbf{u} \cdot \mathbf{e}_i)^2 - \frac{3}{2}\mathbf{u} \cdot \mathbf{u} \right]$$

For D2Q9 we have



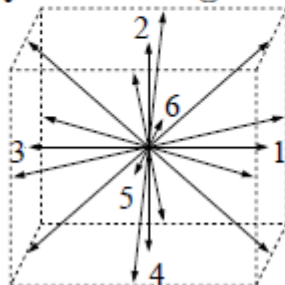
$$w_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i = 1, 2, 3, 4 \\ 1/36 & i = 5, 6, 7, 8 \end{cases}$$

For D3Q15 the weights are:



$$w_i = \begin{cases} 2/9 & i = 0 \\ 1/9 & i = 1 - 6 \\ 1/72 & i = 7 - 14 \end{cases}$$

For D3Q19 the weights are:



$$w_i = \begin{cases} 1/3 & i = 0 \\ 1/18 & i = 1 - 6 \\ 1/36 & i = 7 - 18 \end{cases}$$

Source : [A. Wagner 08] 9

Practical implementation « stream-and-collide »

1. Stream step

$$\tilde{f}_i(x, t) = f_i(x + \mathbf{e}_i, t)$$

2. Collision step

- Compute the moments

$$(\rho, \rho \mathbf{u})(x, t) = \sum_{i \in \mathcal{S}} (1, \mathbf{e}_i) \tilde{f}_i(x, t)$$

- Compute the equilibrium function $\tilde{f}_i^{eq} = \rho w_i [\dots]$
- Collision step :

$$f_i(x, t + \Delta t) = \tilde{f}_i(x, t) + \omega \left[\tilde{f}_i^{eq}(\rho(x, t), \mathbf{u}(x, t)) - \tilde{f}_i(x, t) \right]$$

« ET VOILA ! »

Connection with Navier-Stokes equations (multiscale Chapman-Enskog analysis)

$$\nabla \cdot \mathbf{u} = 0 + O(\Delta t^2),$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p - \nu \Delta \mathbf{u} = O(\Delta t^2 + \Delta t M^2).$$

with

$$\nu = \frac{1}{3}(2\tau - 1).$$

Requires : $u = M \ll 1$.

Artificial EOS:

$$p = p(\rho) = \rho c^2.$$

Von Neuman linear stability
analysis of LBGK scheme:

$$\tau > \frac{1}{2} \quad \Leftrightarrow \quad \omega < 2.$$

Benefits

- Very simple to implement (explicit schemes)
- O(100-lines) Navier-Stokes code with Matlab !
- Wall and fluid boundary conditions OK
- Complex geometries by immersed boundary method (IBM)
- Appears to be suitable for GPU parallel computing
(stream-and-collide procedure)

[Qian, d'Humières, Lallemand 92], [Ginzburg, d'Humières, Krafczyk, Lallemand, 2002], [Chikatamarla, Ansumali, Karlin 06] [Tosi, Ubertini, Succi 06], [Brownlee, Gorban, Levesley 06], [Xu, Luan, Tang, Tao 09]

[Finite Volume interpretation](#) [Dubois, Lallemand 08]

2. Finite Volume method for 2D compressible Euler equations

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0,$$

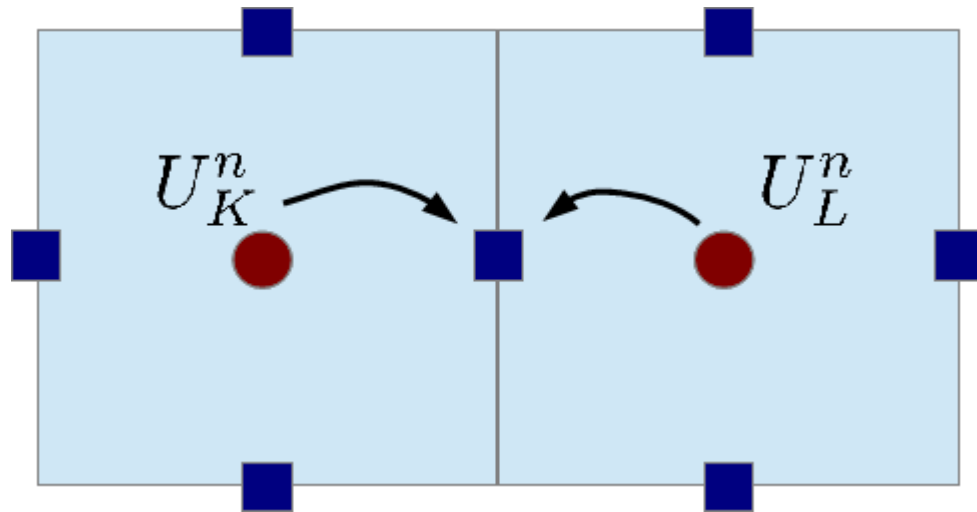
$$\partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = 0,$$

$$\partial_t (\rho E) + \nabla \cdot ((\rho E + p) \mathbf{u}) = 0.$$

$$E = e + \frac{1}{2} |\mathbf{u}|^2, \quad e = \frac{1}{\gamma - 1} \frac{p}{\rho}, \quad \gamma \in (1, 3].$$

- Nonlinear hyperbolic conservative system
- Solutions generally develop discontinuities
- Numerical framework : (stable) finite volumes

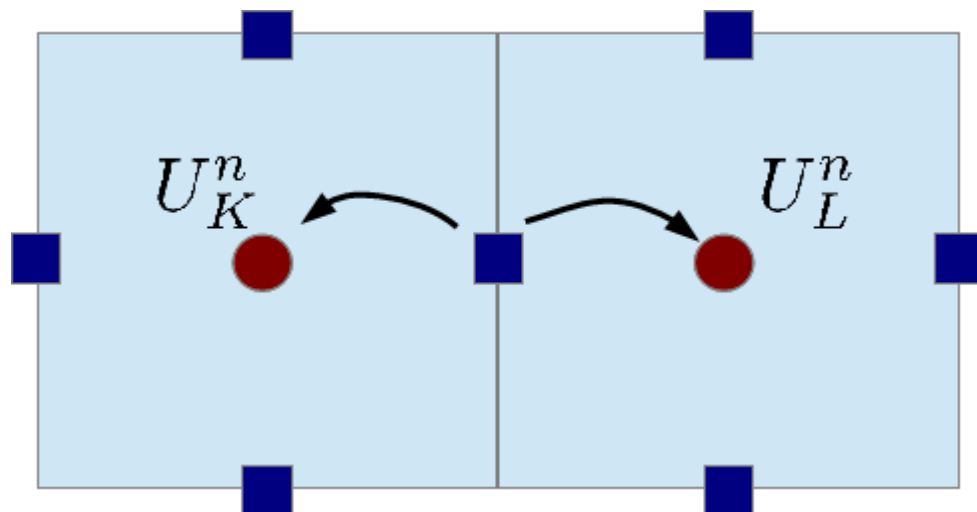
Flux Difference Splitting (FDS) vs Flux Vector Splitting (FVS) methods on GPU



FDS case

- i) **Send** states at interfaces
- ii) Computes numerical fluxes

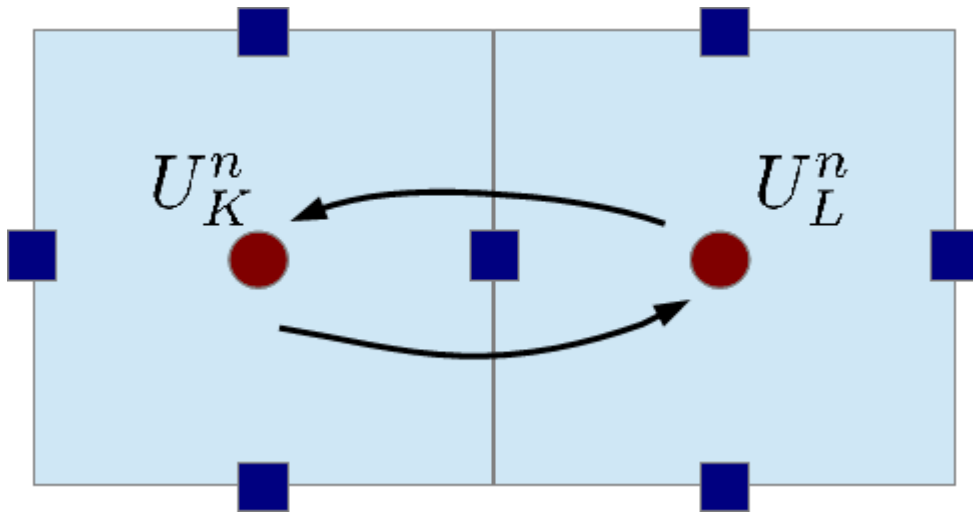
$$\Phi(U_K^n, U_L^n, \nu_{KL}) = \frac{F(U_K^n) + F(U_L^n)}{2} - \frac{1}{2} \int_{\Gamma_{KL}} |A|(s) U'(s) ds$$



- iii) **Send** fluxes at cell centers
- iv) Update states

Flux Difference Splitting (FDS) vs Flux Vector Splitting (FVS) methods on GPU

FVS case $\Phi(U_K^n, U_L^n, \nu_{KL}) = F^+(U_K^n, \nu_{KL}) + F^-(U_L^n, \nu_{KL})$



- i) Compute FVS at cell centers
- ii) **Send** F+ and F- to the neighboring cells
- iii) Update states

→ FVS appears to reduce DRAM communications **by 2**.

Exemple of Flux Vector Splitting

Van Leer's Flux Vector Splitting with Hänel's energy flux correction :

$$F_{\rho}^{+} = \frac{\rho c}{4} (M + 1)^2 1_{(|M| \leq 1)} + \max(u, 0) 1_{(|M| > 1)}$$

$$p^{+} = \frac{p}{4} (M + 1)^2 (2 - M) 1_{(|M| \leq 1)} + p 1_{(M > 1)}$$

$$F_{\rho u}^{+} = u F_{\rho}^{+} + p^{+},$$

$$F_{\rho v}^{+} = v F_{\rho}^{+},$$

$$F_{\rho E}^{+} = H F_{\rho}^{+}, \quad H = E + p/\rho.$$

$$F^{-}(U) = F(U) - F^{+}(U).$$

Quite simple to compute

GPU CUDA implementation

C CUDA kernel prototypes (example)

```
90 //
91 // CUDA kernel prototypes
92 //
93 __global__ void init_kernel (int pitch, var *d_vars);
94 __global__ void setPlotData_kernel(int pitch, var *d_vars, float *plot_data);
95 __global__ void computePrimitiveVariables_kernel(int pitch, var *d_vars, int *solid_data);
96 __global__ void computeFVSx_kernel(int pitch, var *d_vars, flux *fpp, flux *fmm, int *solid_data);
97 __global__ void computeFVSy_kernel(int pitch, var *d_vars, flux *gpp, flux *gmm, int *solid_data);
98 __global__ void updateScheme_kernel(float hx, float hy, float dt,
99     int pitch, var *d_vars, flux *fpp, flux *fmm, flux *gpp, flux *gmm, int *solid_data);
```

Time advance

Very easy !

```
574     for (int i=0; i<FREQ; i++) {
575         //
576         computePrimitiveVariables_kernel<<<grid, block>>>(pitch, d_vars, solid_data);
577         // Compute FVS in x direction
578         computeFVSx_kernel<<<grid, block>>>(pitch, d_vars, fpp, fmm, solid_data);
579         // Compute FVS in y direction
580         computeFVSy_kernel<<<grid, block>>>(pitch, d_vars, gpp, gmm, solid_data);
581         // Apply the Finite Volume scheme
582         updateScheme_kernel<<<grid, block>>>( hx,  hy,  dt, pitch, d_vars,  fpp, fmm, gpp, gmm, solid_data);
583         //
584     } // for int i
```

Data structures : technical aspects

Coalesced memory is a critical keypoint

Q: Arrays-of-Structs (AoS) or Structs-of-Arrays (SoA) ?

$$U = (\rho, \rho u, \rho v, \rho E)^T$$

U[i2d(i,j)].rho = ... // AoS
// Coalesced memory for struct

U.rho[i2d(i,j)] = ... // SoA
// Coalesced memory for array

Option : AoSoA, forced alignment, texture memory, other ?

4. Experience feedback for the PDE community

- Reasonable (suboptimal) speedup (say 10 or 20) is **easy to reach**
- GPU Parallel computing easy for pbs on **cartesian grids**
- Better performance : **tradeoff** to find between code readability and performance.
- Chose **suitable data structures, not too complex** for memory coalescence, read/write transaction: **AoS** vs **SoA** vs **SoAoS**, byte alignment ...
- **Instant interactive computations** made possible, today in 2D, certainly in 3D in 2-3 years...

5. Future works

- 2D air-water 2-fluid flow with interface capturing on GPU
- Lattice Boltzmann for fluid & moving rigid bodies (gas-solid flows)
- Exploration of new suitable paradigm shifts (kinetic formulation of compressible fluids)
- Splitting operator strategies (tuning computations and communications, important keyword)

Acknowledgements

- Institut FARMAN (ENS Cachan) : GPU-powered Alienware laptop support
- NVIDIA Equipement Grant 2011 : 2 TESLA C2070 boards
- FSI-LBM-GPU Farman Project: C. Rey, R. Bennacer, M. Cremonesi, C. Labourdette, F. De Vuyst

Centre de mathématiques et de
leurs applications

CMLA UMR 8536



Cmla

Thank you for your attention