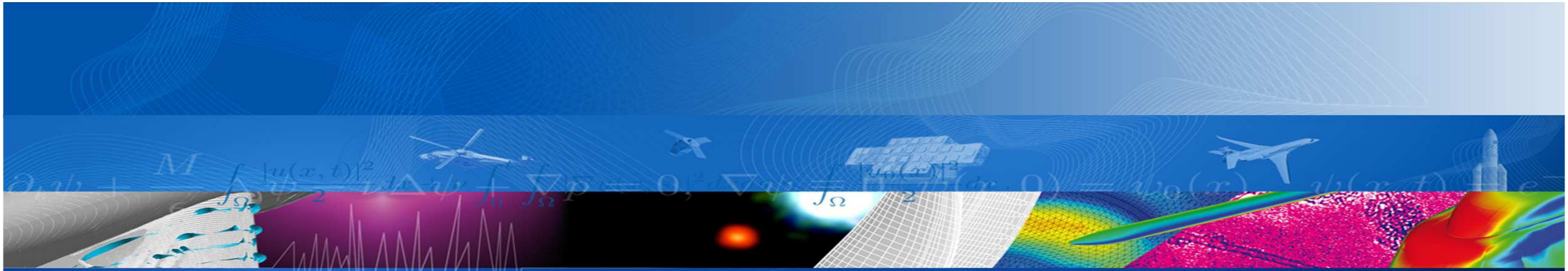


Unsteady CFD for aerodynamics profiles

Jean-Marie Le Gouez, Onera Département MFN

Jean-Matthieu Etancelin, ROMEO

Thanks to Nikolay Markovskiy dev-tech at NVIDIA research Center, GB

ONERA
THE FRENCH AEROSPACE LAB
return on innovation

Unsteady CFD for aerodynamics profiles

- Context
- State of the art for LES / ZDES unsteady Fluid Dynamics simulations of aerodynamics profiles
- Prototypes for new generation flow solvers
- NextFlow GPU prototype : Development stages, data models, programming languages, co-processing tools
- Capacity of TESLA networks for LES simulations
- Performance measurements, tracks for further optimizations
- Outlook

General context : CFD at Onera

The recognized platforms for Research and Industrial applications :

•elsA : aerodynamics and aeroelasticity, in direct and adjoint modes, automatic shape optimization, numerous turbulence models, Zonal Detached Eddy Simulation (ZDES)

•Cedre : combustion and aerothermics,2-phase flows, heat radiation, structural thermics and thermomechanics (Zebulon Onera code)

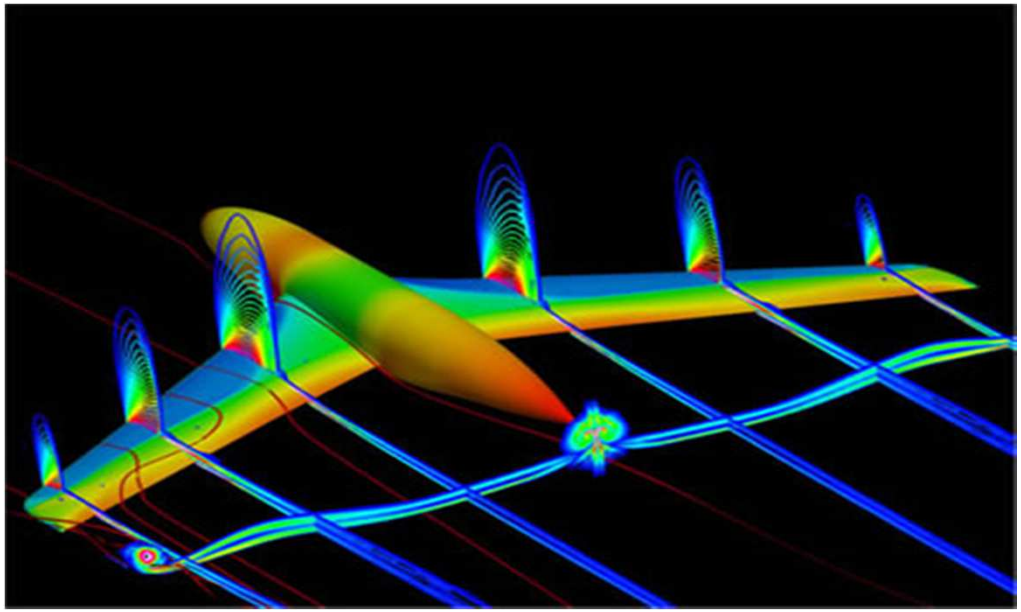
•Sabrina/Funk : unsteady aerodynamics, optimized for LES and aeroacoustics

•Their associated tools for application productivity : links with CAD, overlapping grids, AMR patches, grid deformation tools

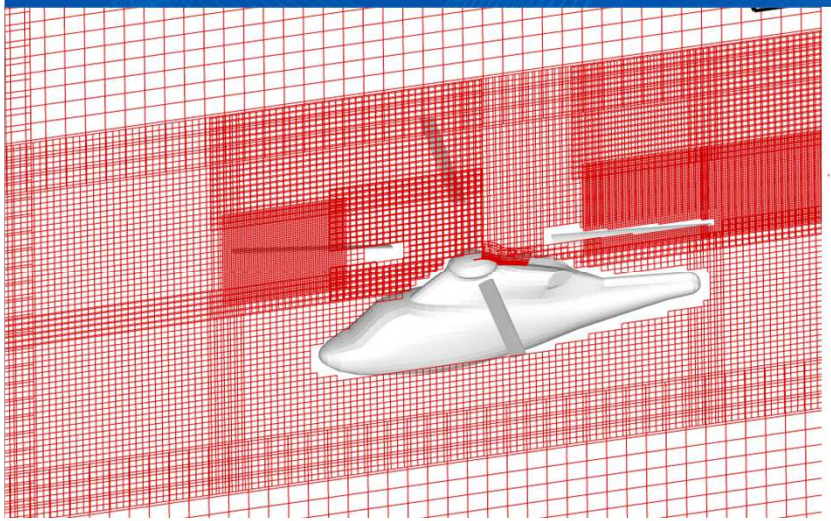
•The multi-physics couplings librairies (collaborations with Cerfacs : Open-Palm)

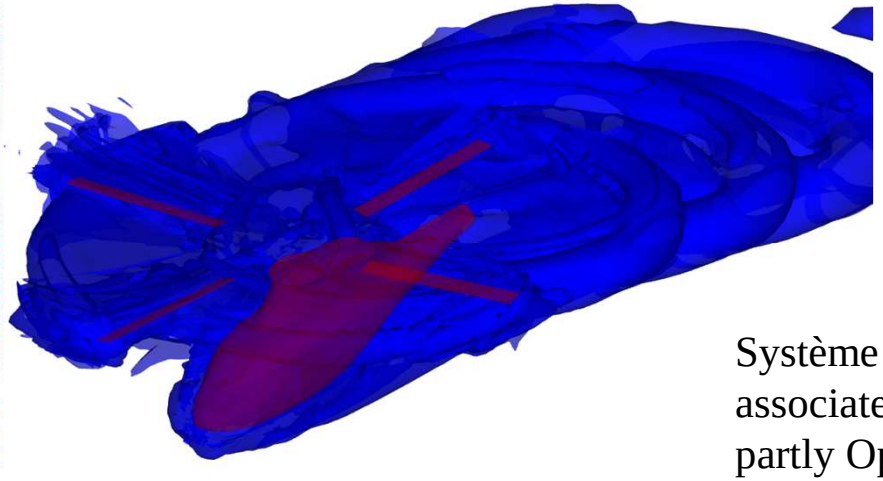
•*Hundreds of thousands of code lines, mix of Fortran, C++, python*

•*Important usage by the aerospace industries*



General context : CFD at Onera





Système Cassiopee
associated to the elsA solver,
partly OpenSource

The expectations of the external users:

•*Extended simulation domains:* ➔ *effects of wake on downstream components blade-vortex interaction on helicopters, thermal loadings by the reactor jets on composite structures,*

•*Model of full systems and not only the individual components : multi-stage turbomachinery internal flows, couplings between the combustion chamber and the turbine aerodynamics, …*

•*More multi-scale effects : representation of technological effects to improve the overall flow system efficiency : grooves in the walls, local injectors for flow / acoustics control,*

•*Advanced usage of CFD : adjoint modes for automatic shape optimization and grid refinement, uncertainty management, input parameters defined as pdf,*

CFD at Onera

Hard to conduct a deep reengineering of the existing solvers (important workload on the improvement of physical modelling with them)

Risks on their future scalability towards PetaFlops computing (expected within 3-5 years)

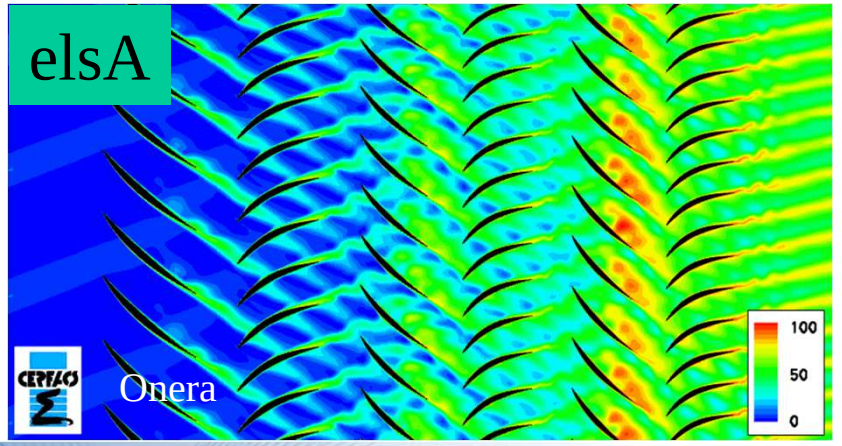
Development of prototypes is undertaken on the following subjects, for more modular solvers and couplers, optimized in their field :

Grid strategies :
- hybrids structured / unstructured
- overlapping, refined automatically,
- high order grids (curved faces),

Numerical schemes and non-linear solvers :
- Discontinous Galerkine Method (AGHORA project)
- High order Finite Volumes (NextFlow project)

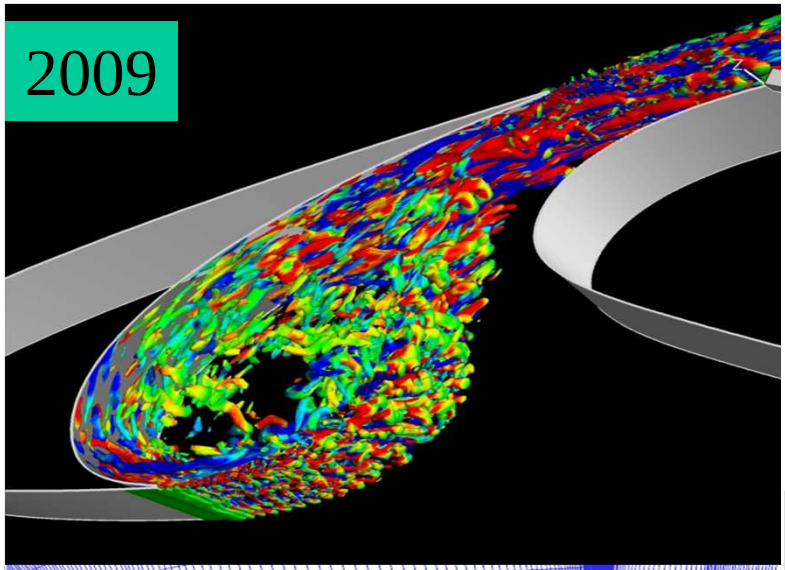
Multi-physics and multi-model couplings : CWIPI library, OpenPalm with Cerfacs

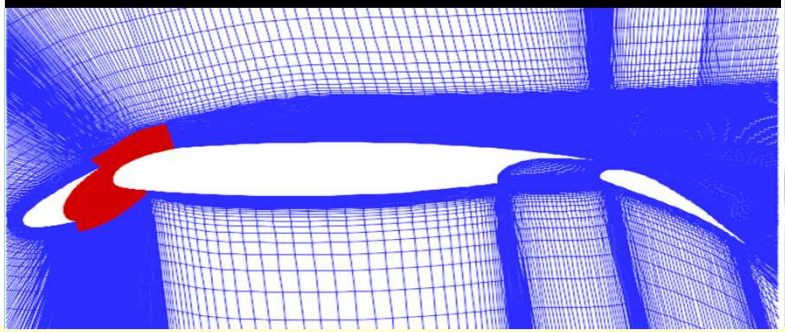
HPC : MPI / OpenMP, Vectorization on CPU (X86)
Heterogeneous Hardware architectures CPU / GPU



Improvement of predictive capabilities over the last 5 years : RANS / zonal LES of the flow around a High-Lift deployed wing

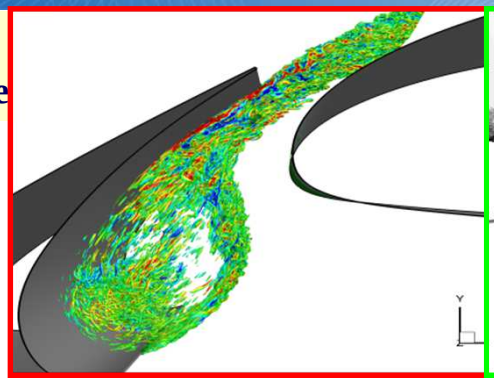
2009

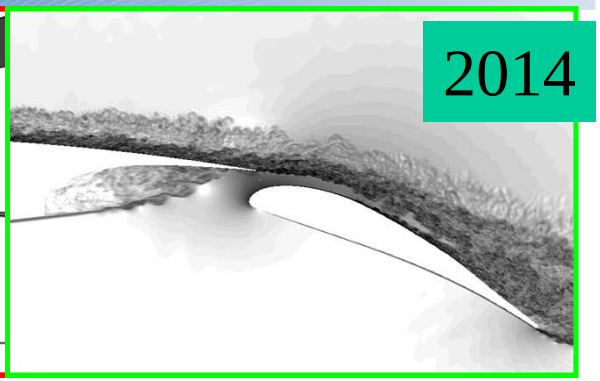




2D steady RANS and
3D LES 7,5 Mpts

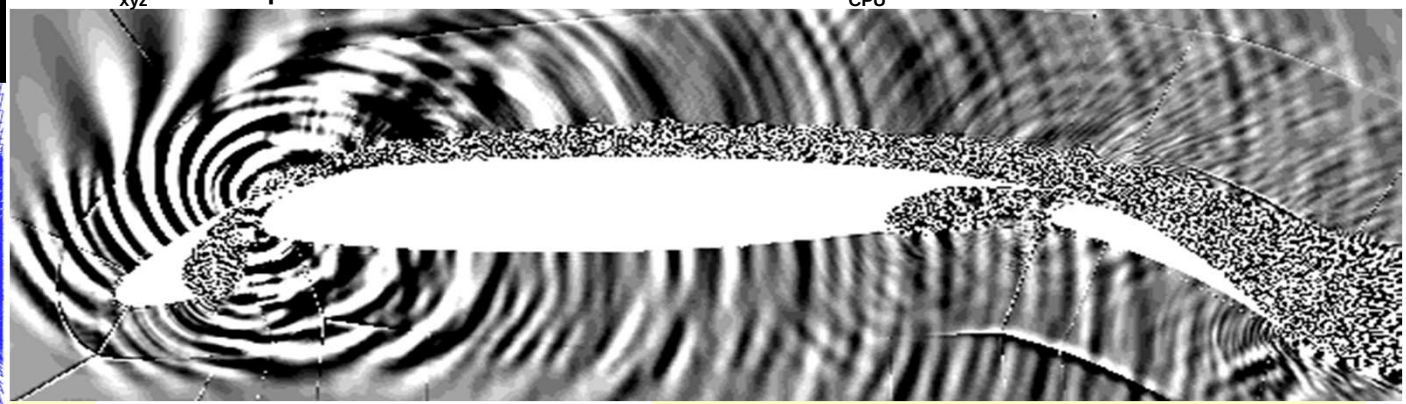
Mach 0.18
Rey 1 400 000/corde





2014

- ***Optimized on a CPU architecture MPI / OpenMP / vectorization***
- **CPU ressources for 70ms of simulation : JADE computer (CINES)**
- **N_{xyz}~ 2 600 Mpts 4096 cores / 10688 domains T_{CPU}~ 6 200 000 h**

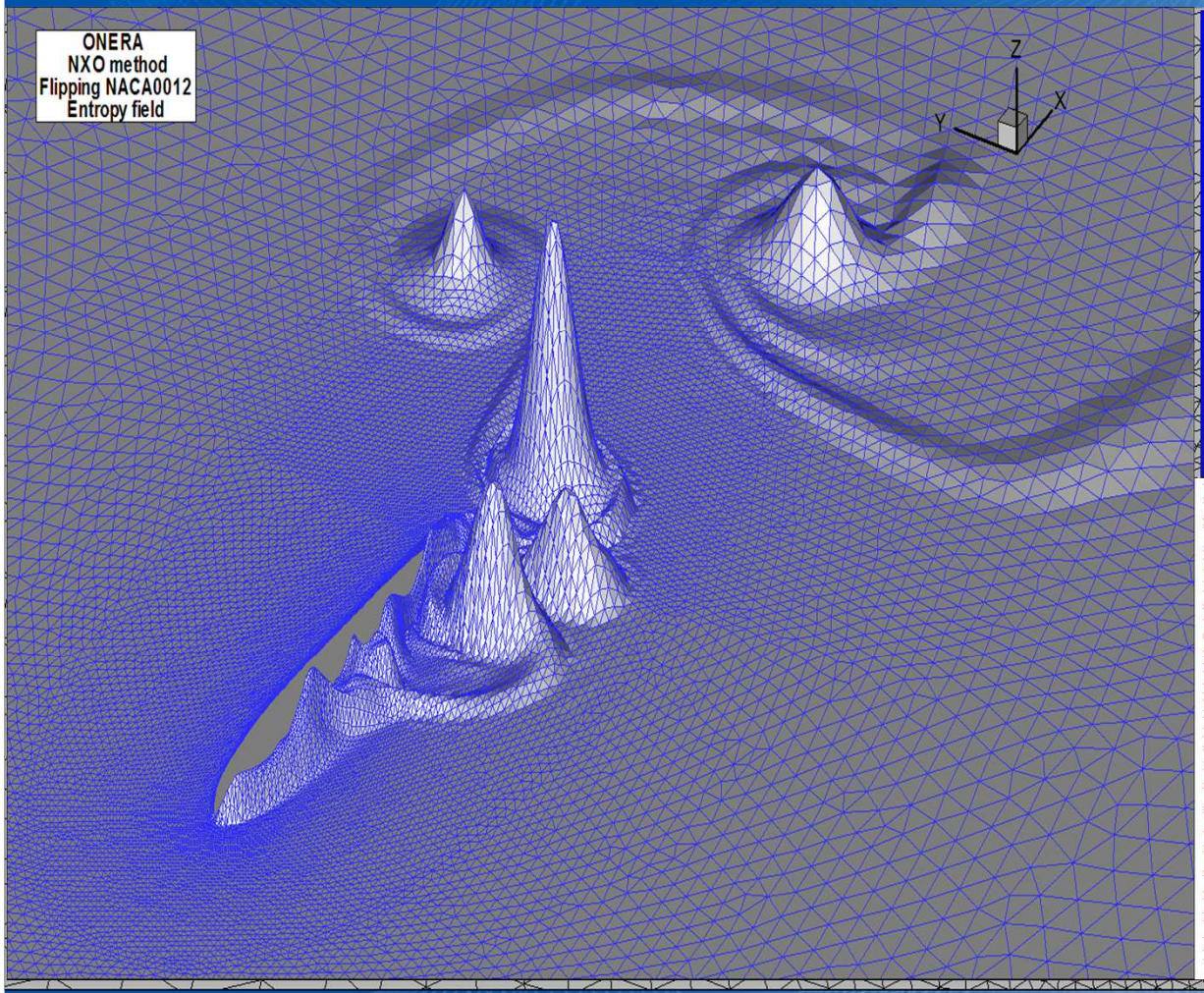


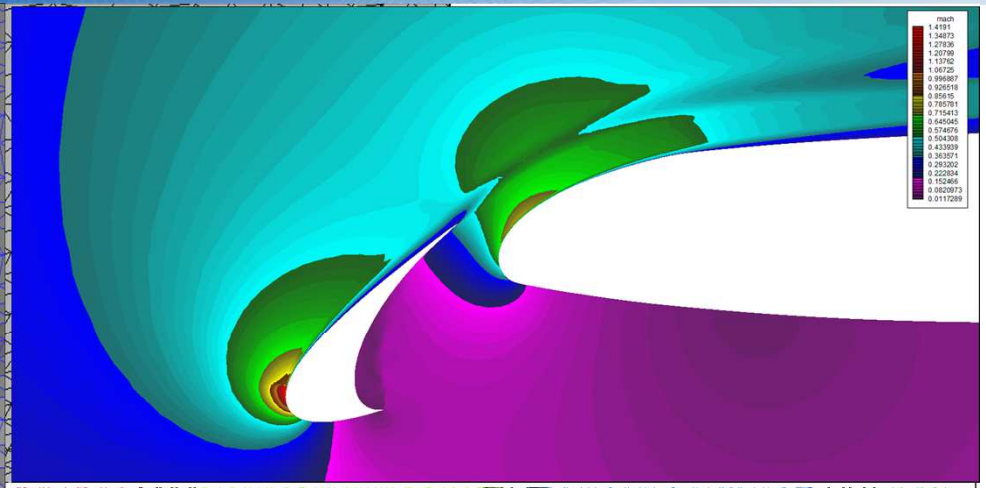
- ***Computed field of dilatation***
- ***Identification of acoustic sources***

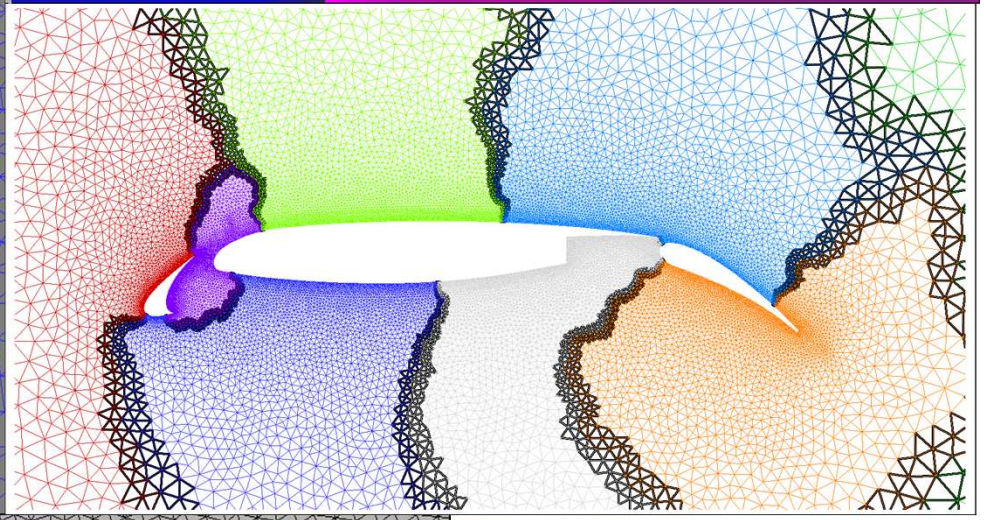
LEISA project DLR / Onera cooperation
ONERA FUNK software

NextFlow : HO Finite Volume for RANS / LES

Demonstration of the feasability of porting these algorithms on heterogeneous architectures







ONERA
THE FRENCH AEROSPACE LAB

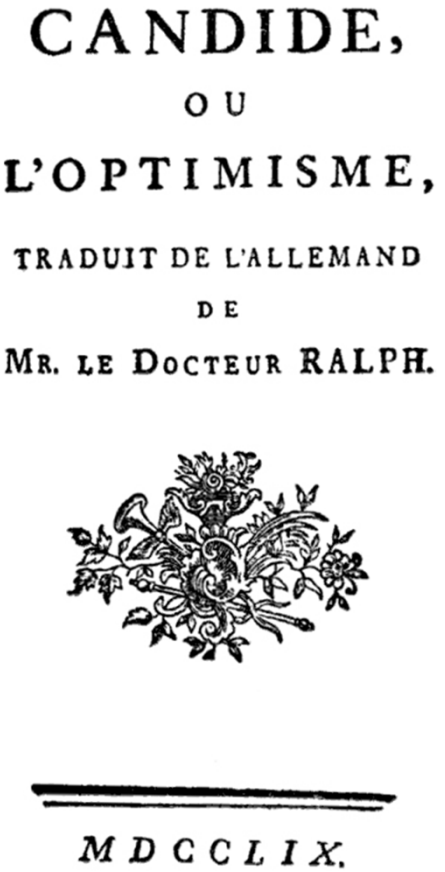
Starting from a Fortran / MPI base, with a management of a complex data model on partitioned grids, porting both on OpenMP / Fortran and C / Cuda, sharing an interface that does a transpose of work arrays, manages pointers on CPU and GPU memories,...

Preliminary work : I,j,k structure of the grid and the arrays, choice of the programming model (shared or distributed memory, acceleration by directives or dedicated language ➔ choice of C / Cuda / structures

1st version : on the fully unstructured grid. Tests of fine partitioning relying on advanced cache usage on the GPU,

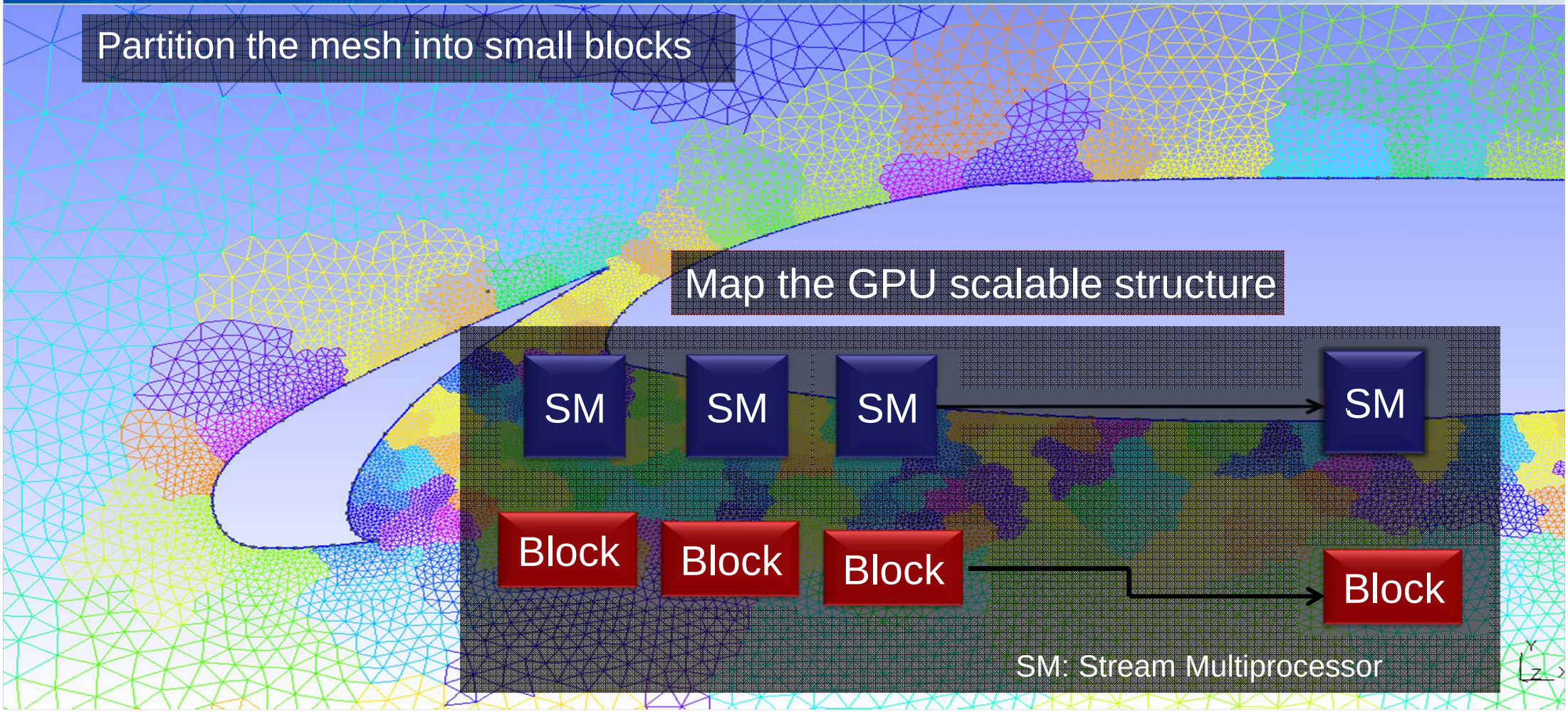
2nd version : hierarchical model, with an initial coarse grid which is dynamically refined on the GPU : this ensures a coalescent data access to global memory,

3rd version (on-going work) : 2.5 D model, periodic in spanwise direction, automatic vector processing on the CPU, and full data parallel model for the GPU in the homogeneous 3rd dimension


CANDIDE,
OU
L'OPTIMISME,
TRADUIT DE L'ALLEMAND
DE
MR. LE DOCTEUR RALPH.
MDCCLIX.

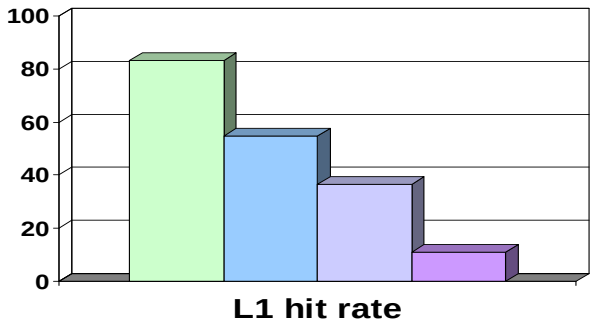
« Cela est bien dit, répondit Candide, mais il faut écrire notre Cuda »

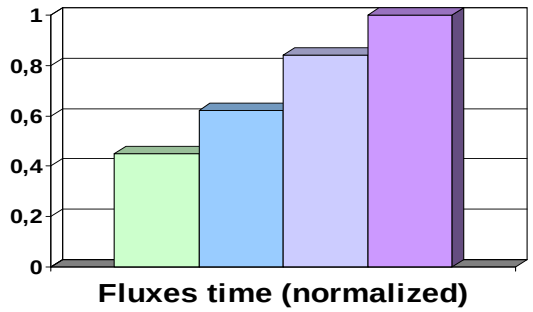
1st Approach: Block Structuration of a Regular Linear Grid

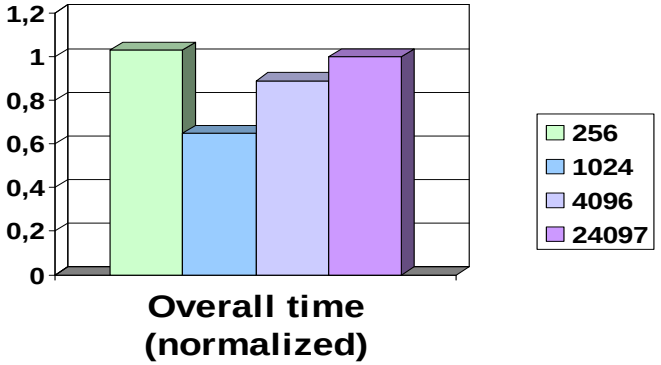


Advantage of the Block Structuration

- Bigger blocks provide
  - Better occupancy
  - Less latency due to kernel launch
  - Less transfers between blocks
- Smaller blocks provide
  - Much more data caching







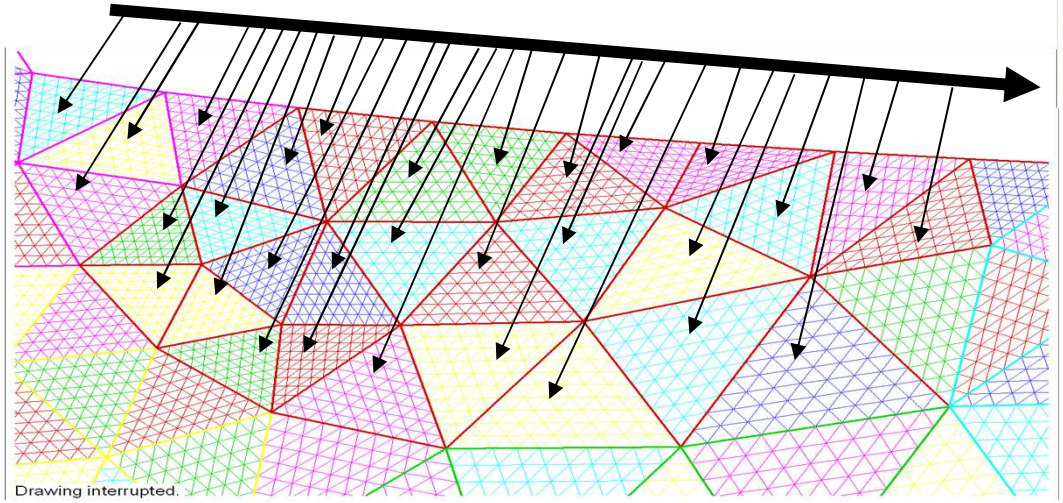
- **Final speedup wrt. to 2 hyperthreaded Westmere CPU: ~2**

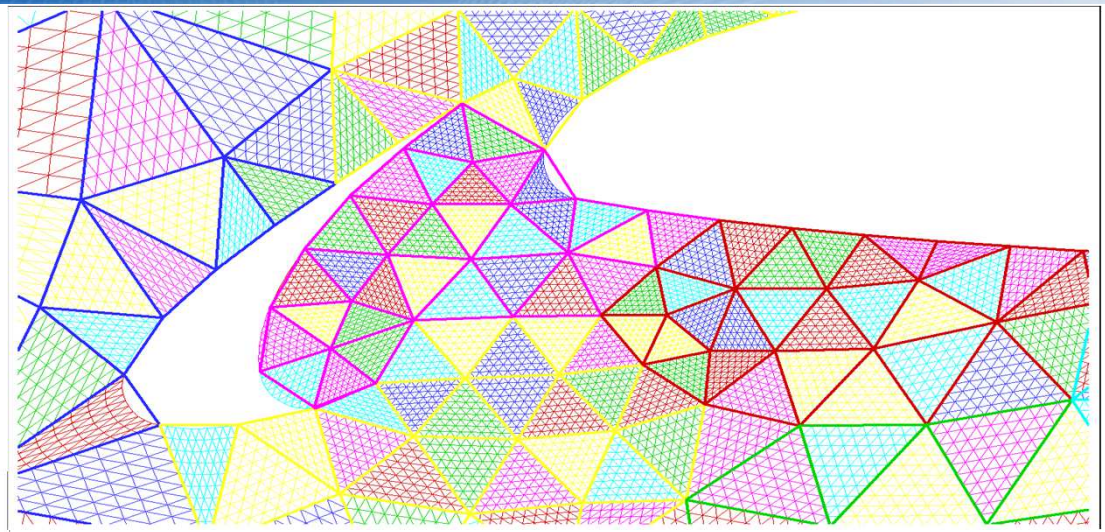
NXO-GPU Phase 2 : Imposing a sub-structuration to the grid and data model
(inspired by the 'tesselation' mechanism in surface rendering)

Unique grid connectivity for the algorithm

Optimal to organize data for *coalescent memory access* during the algorithm and communication phases

Each coarse element in a block is allocated to an inner thread (threadId.x)



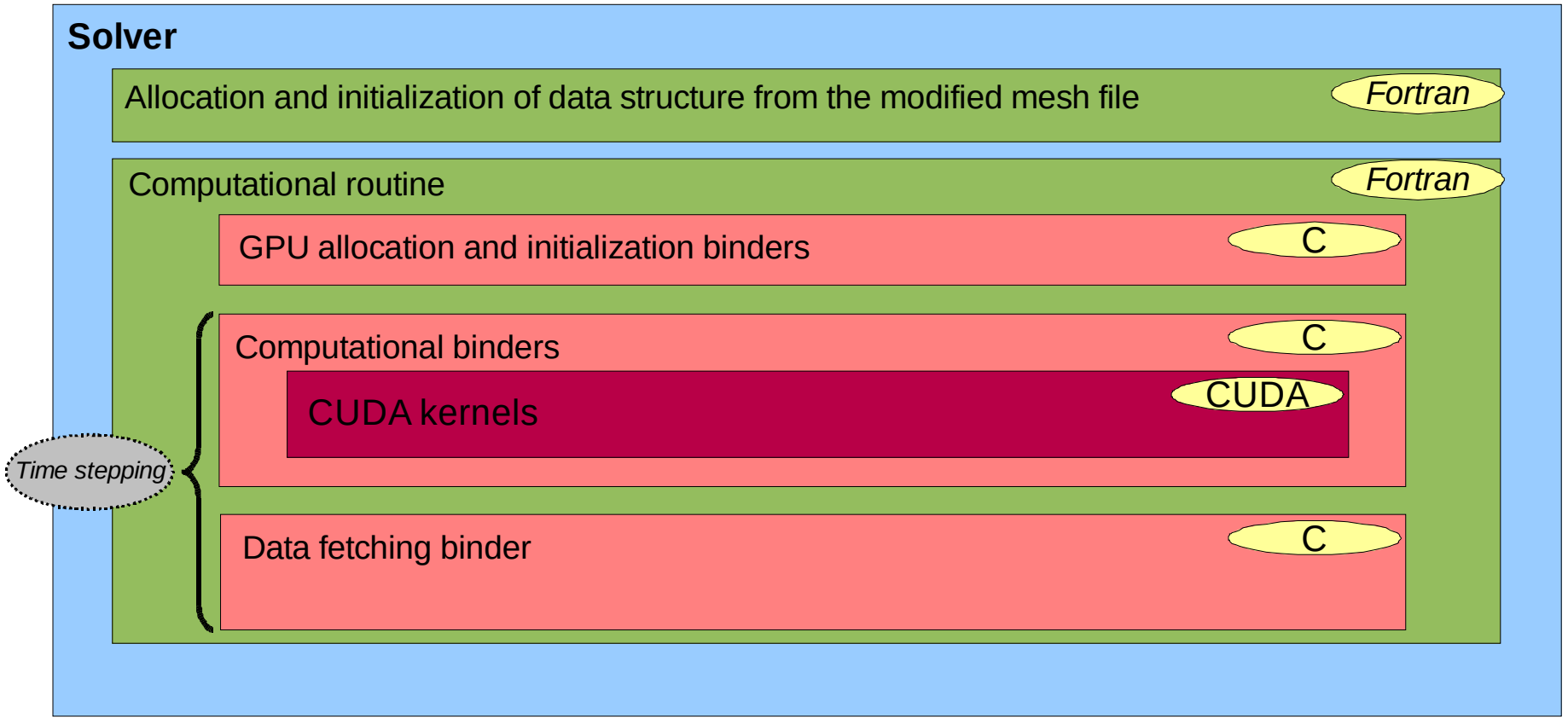


Hierachical model for the grid : high order (quartic polynomial) triangles generated by gmsh refined on the GPU

the whole fine grid as such could remain unknown to the host CPU

Code structure

Preprocessing
Mesh generation and block and generic refinement generation



Postprocessing
Visualization and data analysis

Version 2 : Measured efficiency on Tesla 2050 and K20C (with respect to 2 Cpu Xeon 5650, OMP loop-based)

Results on a K20C : Max. Acceleration = 38 wrt to 2 Westmere sockets

Improvement of the Westmere CPU efficiency : OpenMP task-based ➔ the blocks are refined on the CPU also, then the K20C GPU / CPU acceleration drops to 13 (1 K20c = 150 Westmere cores)

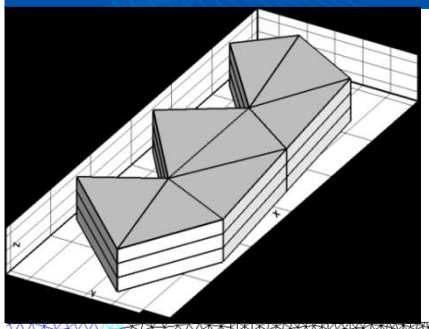
In fact this method is memory bounded, and GPU bandwidth is critical. More CPU optimisation needed (cache blocking, vectorisation ?)

Flop count : around 80 Gflops/K20C

These are valuable flop, not Ax=b flop, but Riemann solver flop with high order (4th, 5th) extrapolated values, characteristic splitting, … : requires a very high memory traffic to permit theses flops

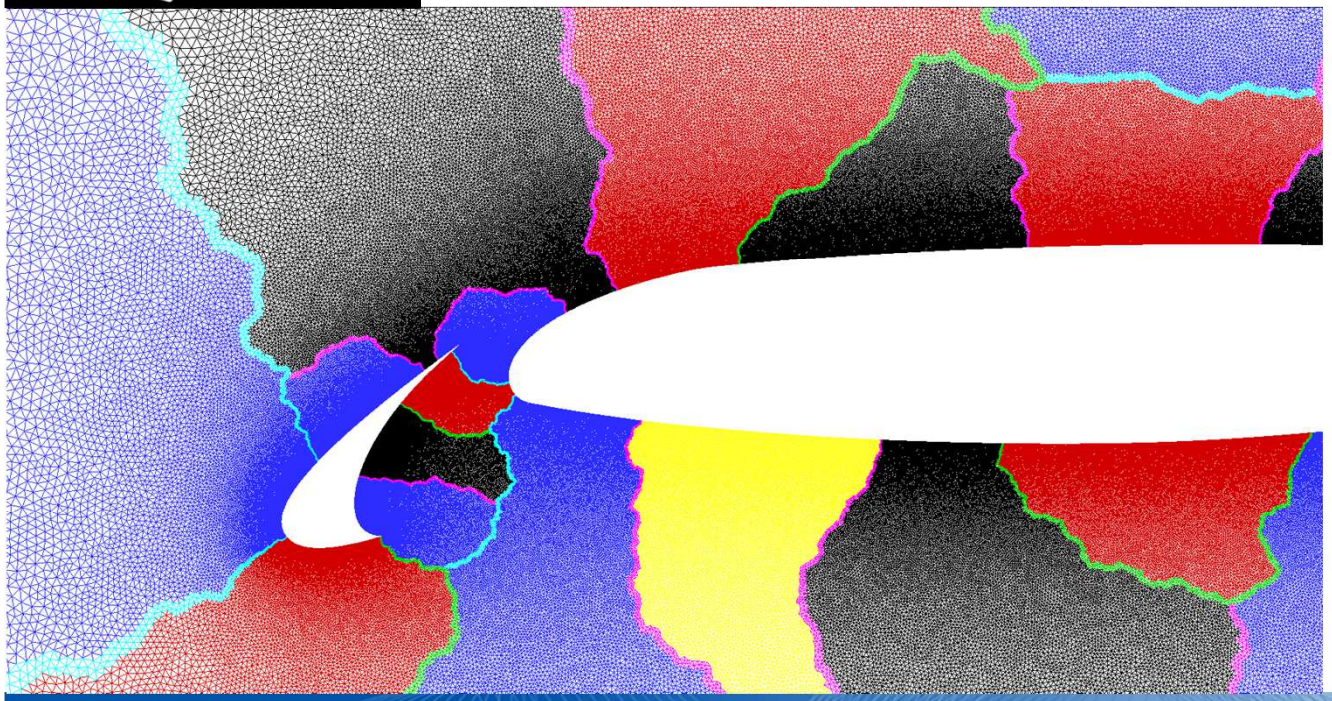
Thanks to the NVIDIA dev-tech department for their support, “ my flop is rich”

Version 3 : 2.5D periodic spanwise (cshift vectors), MULTI-GPU / MPI



High CPU vectorisation (all variables are vectors of length 256 to 512) in the 3rd homogeneous direction

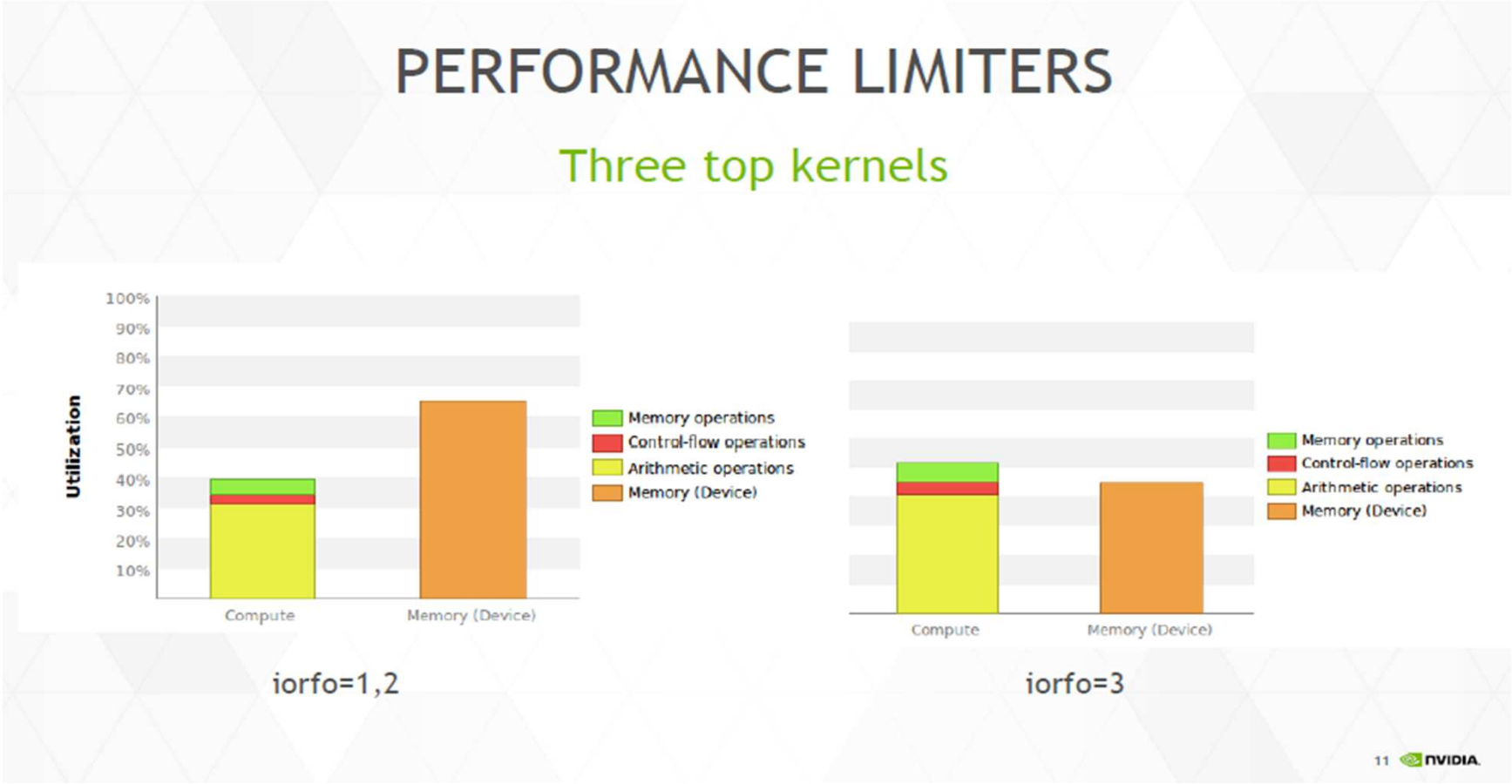
Full data parallel Cuda kernels coalesced



Objective : one Billion cells on a cluster with 64 TESLA K20 (40 000 cells * 512 spanwise stations per partition)

The CPU (Fortran) and GPU (C/ Cuda) versions are in the same executable, for efficiency and accuracy comparisons

Version 3 : 2.5D periodic spanwise (cshift vectors), MULTI-GPU / MPI



Version 3 Initial Kernel Optimization (done by NVIDIA DevTech Great-Britain)

RESULTS

Comparison with Original code (1 partition)

Original		Current optimizations	
Compute :	85.0936872959137 12.8738024234772	Compute :	84.5988802909851 5.95152688026428
Newtime :	2.962398529052734E-002 3.664016723632812E-003	Newtime :	2.988314628601074E-002 3.657817840576172E-003
Newiter :	0.201013803482056 2.434015274047852E-002	Newiter :	0.196562051773071 3.784275054931641E-002
Dtloc :	7.01542568206787 0.916594982147217	Dtloc :	7.00739192962646 0.480051517486572
Timeres :	1.90137290954590 0.292651176452637	Timeres :	1.89031934738159 0.300009727478027
Fluxes :	69.2380857467651 9.93227958679199	Fluxes :	68.8473780155182 4.28139758110046
Fluxbal :	3.26401877403259 0.913108825683594	Fluxbal :	3.21450090408325 0.439968824386597
Update :	3.44401168823242 0.791139841079712	Update :	3.41265749931335 0.408583641052246
Messages :	1.347064971923828E-004 2.384185791015625E-005	Messages :	1.873970031738281E-004 1.502037048339844E-005

8 IVB cores vs K40(ECC ON, GPU Boost ON)

12 NVIDIA

CPU time

Left column : IvyBridge (8 OMP threads)

Right column : K40

Full time steps and 7 main kernels

Performance strategy : Increase occupancy, reduce registers' use, reduce amount of operations with global memory

After some GPU optimizations : Acceleration 14 on one K20c with respect to 8 IVB cores, GPU memory Bandwith 150 GB/s

For LES, GPU memory size not too much of a problem : 20 million cells stored on a K20 (6 Gbytes), 40 millions on a K40

Version 3 : on-going work

Improve the efficiency of the communications between partitions : MPI / Cuda, GPUDirect,

Overlap communications with computations at the center of the partitions,

Verify the scalability curve from 1 to 128 accelerators,

Choose a strategy for the CPU usage :

- **identify algorithmic phases that are less pertinent for GPU,**
- **specialize the CPU for co-processing tasks : Spatial Fourier transforms, management and computation of mesh refinement indices, preparation of cell metrics for the next time step for moving/deforming grids, computation of iso-value surfaces**

Compare different versions of the Fluxes and Riemann solvers, which require less registers,

Compare the efficiency and accuracy to standard Fortran / MPI 2nd order codes, while using larger mesh size

Prepare for the specification of the next generation of CFD solvers