Mercator Research Center Ruhr
Eine Initiative der Stiftung Mercator und der Universitätsallianz Ruhr

technische universität dortmund

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

CCSS

VINDDkanaltunnel

A virtual, interactive numerical wind-tunnel

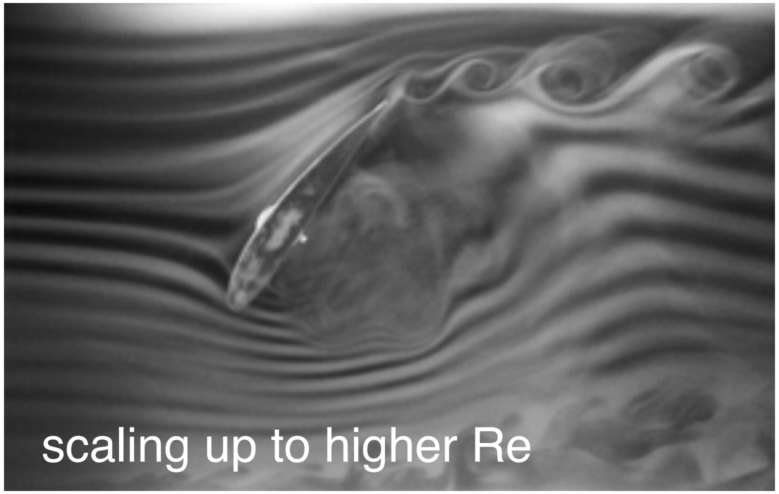
with GPU acceleration, real time visualization and real time interaction

Thuong Nguyen, Tom Fogal, Raphael Münster
Andreas Kempf, Jens Krüger, Stefan Turek

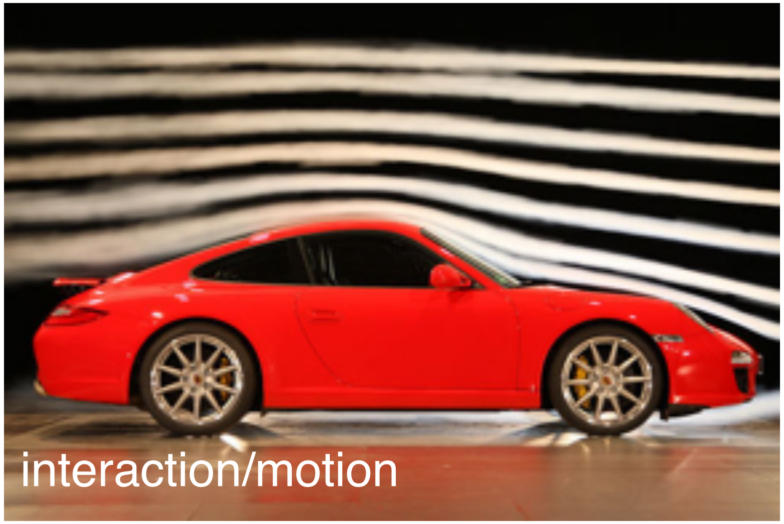
Lehrstuhl Fluiddynamik, Institut für Verbrennung und Gasdynamik, Universität Duisburg-Essen
Lehrstuhl III für Mathematik, Institut für Angewandte Mathematik, Technische Universität Dortmund
Lehrstuhl für Hochleistungsrechnen, Universität Duisburg Essen

Motivation

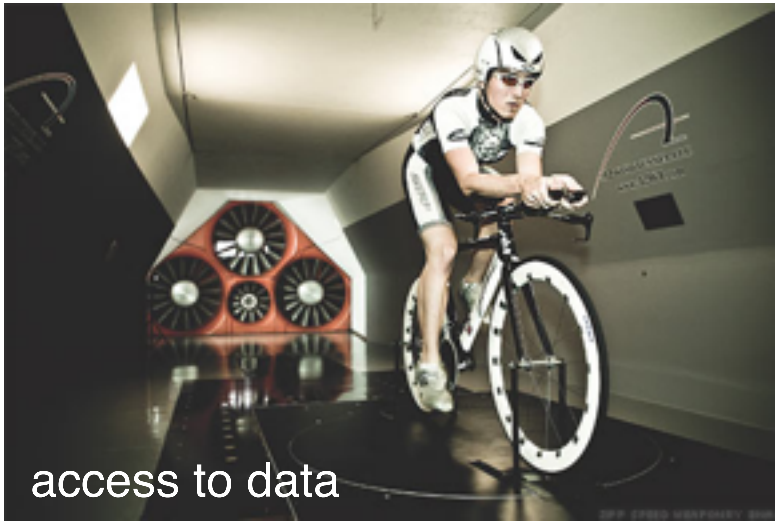
Supersonic Wind Tunnel - Arnold Engineering Development Center, Tennesee, 1960



Fog (water particles) wind tunnel visualization of a NACA 4412 airfoil at a low Re=20.000

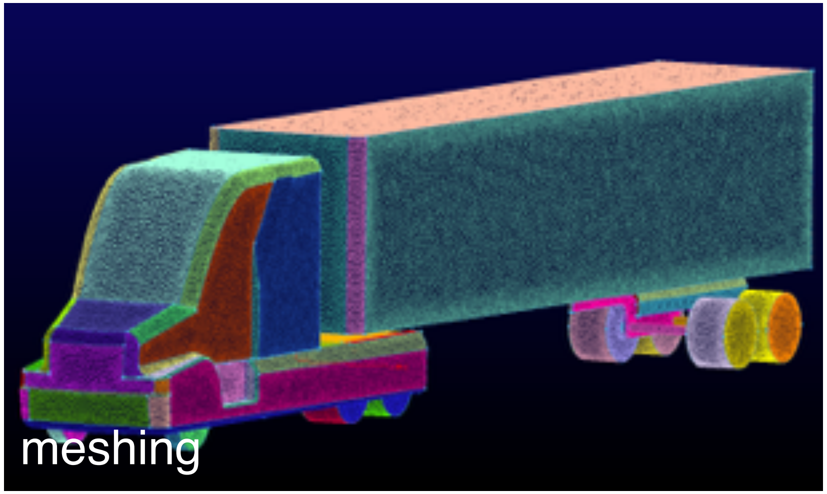


Car in a windtunnel



Testing of bicycle rims

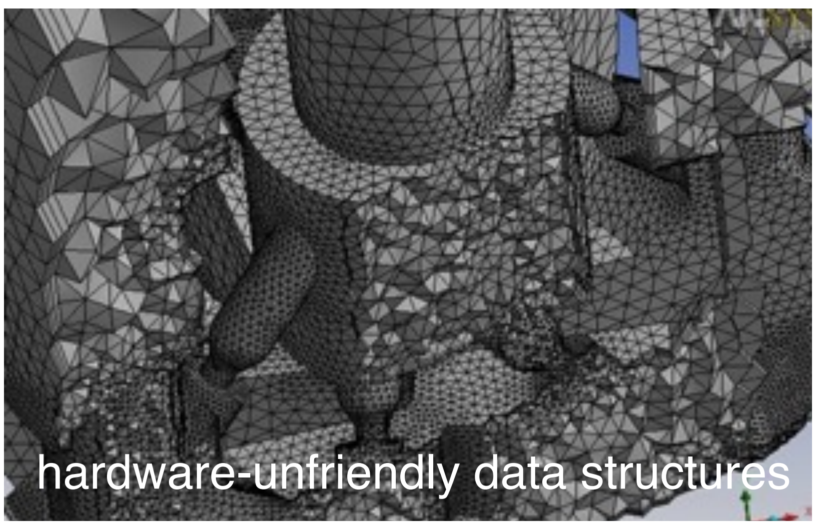
Motivation



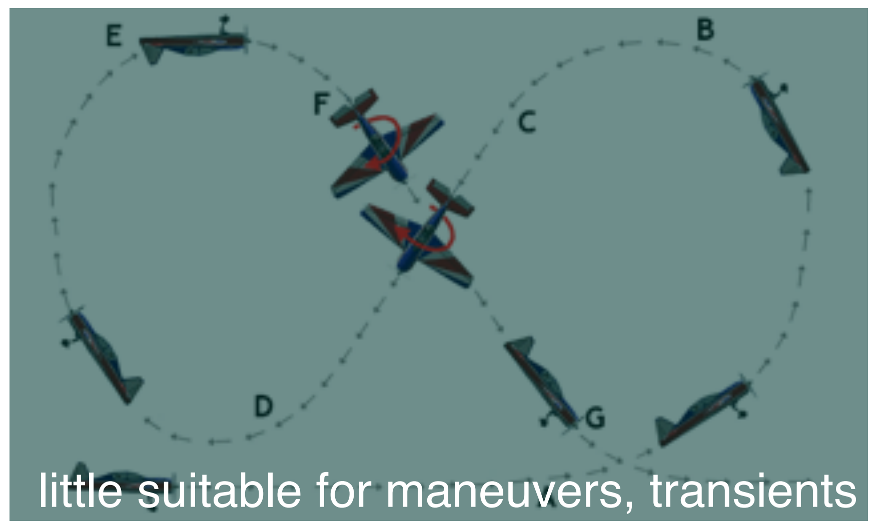
meshing

```
24792:   Det = 0.67364E-05   Tprobl = 0.18041E+00
24793:   Det = 0.67215E-05   Tprobl = 0.18042E+00
24794:   Det = 0.69165E-05   Tprobl = 0.18043E+00
24795:   Det = 0.74239E-05   Tprobl = 0.18043E+00
24796:   Det = 0.74878E-05   Tprobl = 0.18044E+00
24797:   Det = 0.70956E-05   Tprobl = 0.18045E+00
24798:   Det = 0.69537E-05   Tprobl = 0.18045E+00
24799:   Det = 0.70583E-05   Tprobl = 0.18046E+00
MTV OUTPUT        (TVWrap)
24800:   Det = 0.73700E-05   Tprobl = 0.18047E+00
24801:   Det = 0.71020E-05   Tprobl = 0.18048E+00
```

missing interactivity
(results are known late)



hardware-unfriendly data structures



little suitable for maneuvers, transients

Numerical Wind Tunnel - 90ies

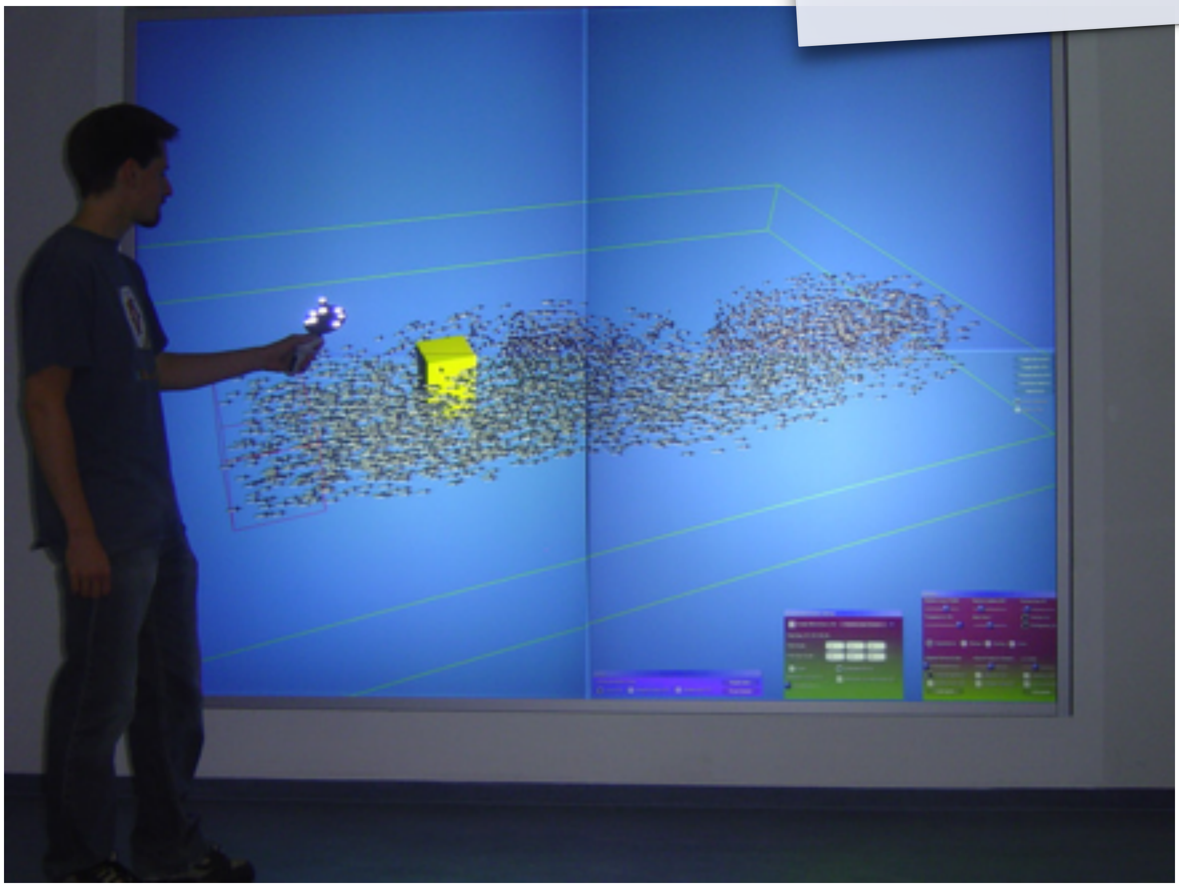
Numerical Wind Tunnel - Japan
TOP500 1993-1995

Processing elements (PE) nodes	166
Control processor (CP) nodes	2
Network	Two 421 MB/s crossbar networks
PE memory	256 MB
Total performance	280gigaFLOPS 45GB
Programming languages	Fortran, C
Parallel libraries	PVM, PARMACS, MPI

Virtual Wind Tunnel - 00ies

Numerical Wind Tunnel - TUM 2005

- Interactive visualisation of flow-fields
- Focus on post-processing



Virtual Wind Tunnel - 10^s

- *Interaction?*
- *Instantaneous meshing?*

The Vision

- Build a virtual wind tunnel
- Achieve fast CFD solver using parallel GPU programming
- Interact with the simulation in real time
- Interact with real wind tunnel?
- Receive feedback immediately via real time visualization
- Represent complex geometries in simple manner

Governing Equations

Filtered Navier-Stokes equations:

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial \bar{\rho}\widetilde{u_j}}{\partial x_j} = 0$$

$$\frac{\partial \bar{\rho}\widetilde{u_i}}{\partial t} + \frac{\partial \bar{\rho}\widetilde{u_j}\widetilde{u_i}}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\bar{\rho}\widetilde{\nu}\left(\frac{\partial \widetilde{u_i}}{\partial x_j} + \frac{\partial \widetilde{u_j}}{\partial x_i}\right) - \frac{2}{3}\bar{\rho}\widetilde{\nu}\frac{\partial \widetilde{u_k}}{\partial x_k}\delta_{ij} - \bar{\rho}\tau_{ij}^{sgs}\right]$$

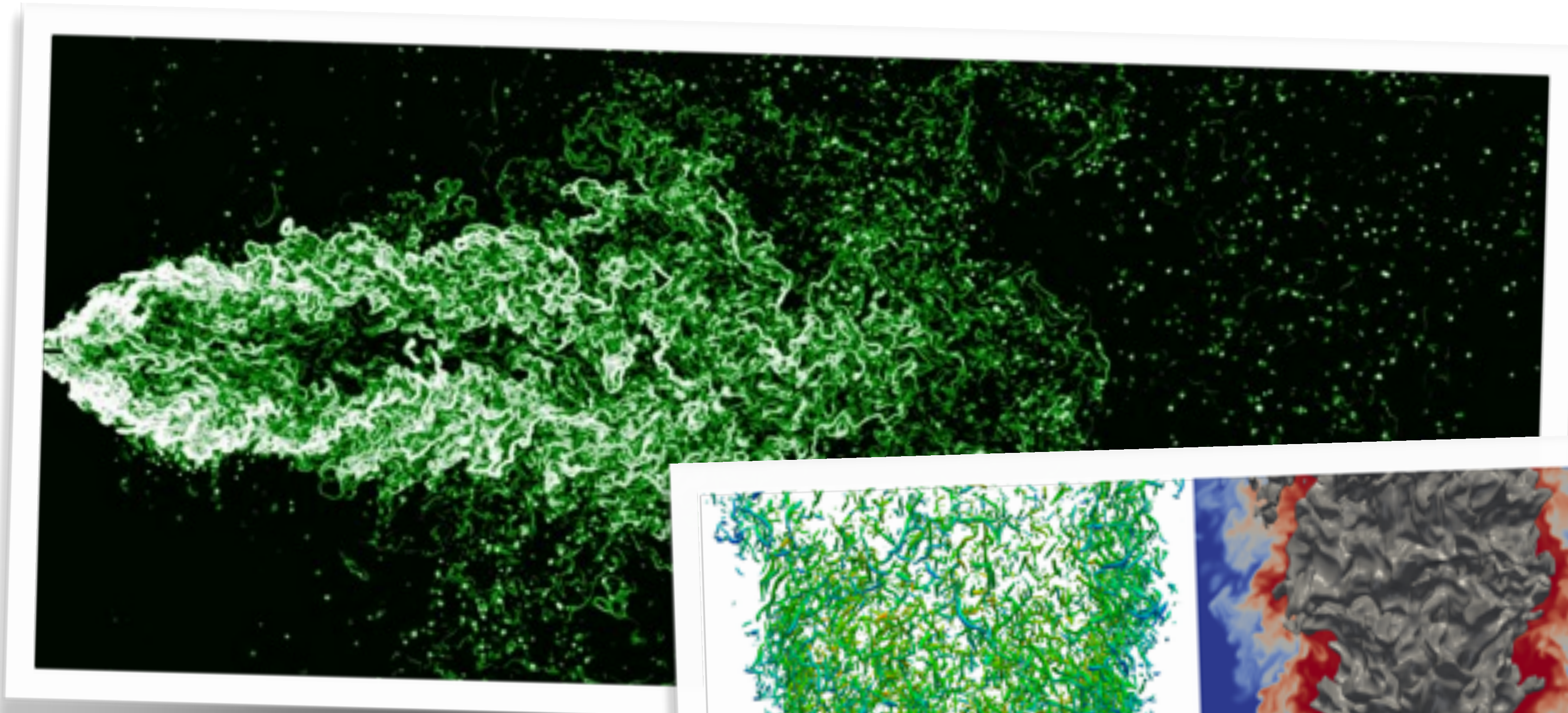
$$\frac{\partial \bar{\rho}\widetilde{e}}{\partial t} + \frac{\partial \left(\bar{\rho}\widetilde{e}\right)\widetilde{u_j}}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\kappa\frac{\partial \widetilde{T}}{\partial x_j}\right) + \frac{\partial}{\partial x_j}\left(\widetilde{u_i}\widetilde{\sigma_{ij}}\right) - \frac{\partial}{\partial x_j}\left(\kappa_{ij}^{sgs}\right)$$

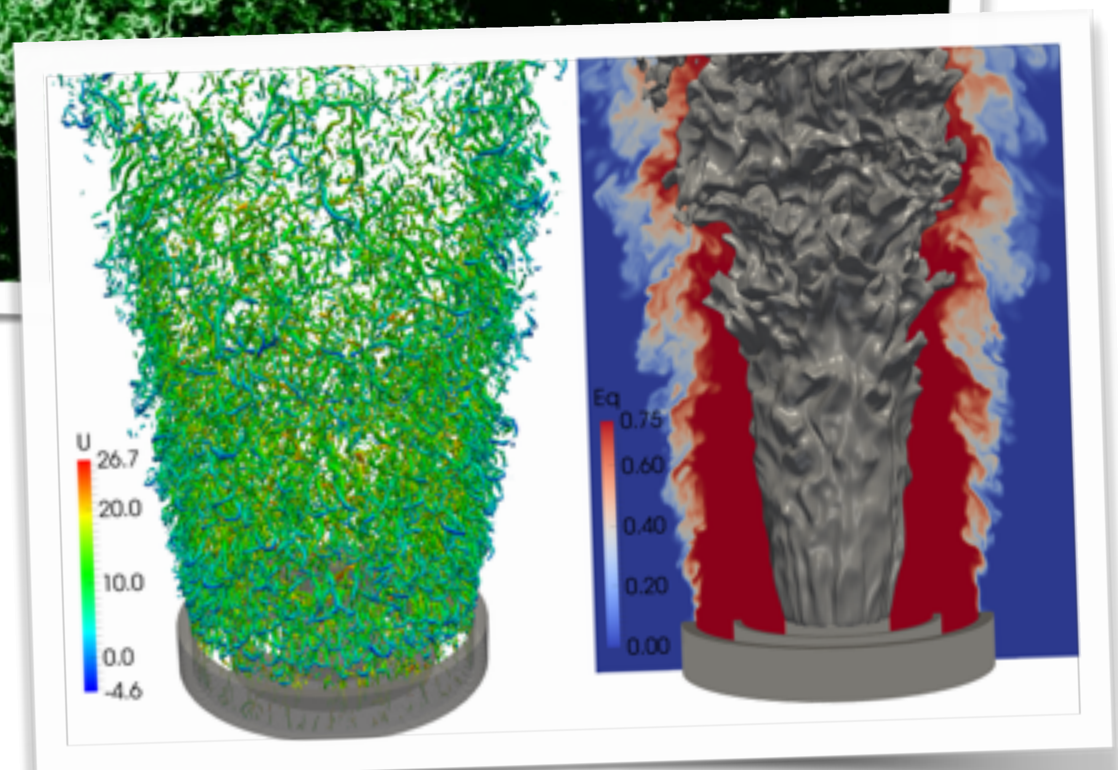
$$\frac{\partial \bar{\rho}\widetilde{\phi}}{\partial t} + \frac{\partial \bar{\rho}\widetilde{\phi}\widetilde{u_i}}{\partial x_i} = \overline{\frac{\partial}{\partial x_i}\left(\rho D\frac{\partial \phi}{\partial x_i}\right)}$$

The equation of state

$$p = \rho R T \, , c_v = \frac{R}{\gamma - 1} \, , c_p = c_v + R$$

CFD Solver - PsiPhi





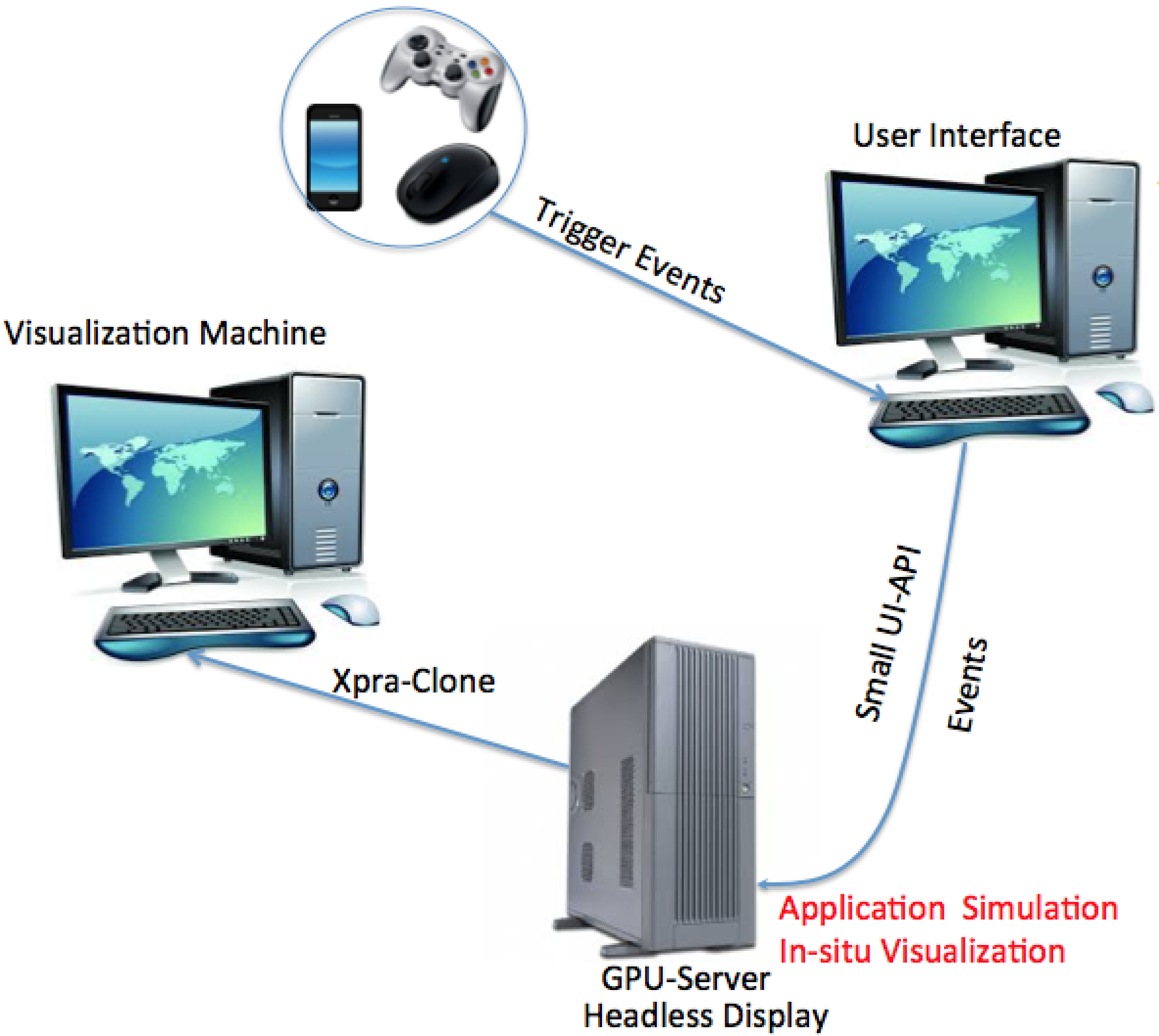
scalable from PC to vHPC
LES, DNS
coal, spray, gas flames
radiation
(in-) compressible
moving geometries
10^1-10^2 faster than unstructured

CFD Solver - PsiPhi

Grid	Equidistant Cartesian grids
Mesh motion	Immersed boundaries-Lagrangian particles
Flow solver	Fully compressible (explicit)
Inlet - Outlet **Top & Bottom** **Left & Right**	NSCBC at inlet and outlet Wall bounded BC Periodic BC
Time stepping **Convection** **Diffusion**	Explicit O(<3) low storage RK TVD CDS

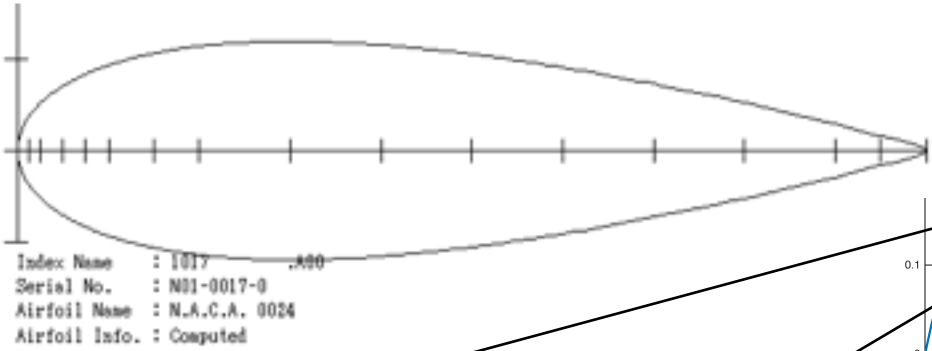
```
SUBROUTINE FlxAddCDS(Field, Vel, Flux) !###############                              ###########
   !---------- modules ----------------------------                               -----------
      USE pnet; IMPLICIT NONE
   !---------- variables --------------------------                               -----------
      REAL,INTENT(IN)    :: Field(Imap2, Jmap2, Kmap2)
      REAL,INTENT(IN)    :: Vel(Imap2,Jmap2,Kmap2,Dma)                   ! Velocities
      REAL,INTENT(INOUT) :: Flux(Imap2,Jmap2,Kmap2,Dma)                  ! Fluxes
   !------------------------------------------------------------------------------------------
      ! FlxAddCDS adds convective flux from central differencing (CDS) to flux.
   !__________ BEGIN CODE ____________________________________________________________________
      Flux(Ifim:Ila,Jfim:Jlap,Kfim:Klap,1) = Flux(Ifim:Ila,Jfim:Jlap,Kfim:Klap,1) &
                                           + Vel(Ifim:Ila,Jfim:Jlap,Kfim:Klap,1) &
                                            *0.5*(Field(Ifim:Ila,Jfim:Jlap,Kfim:Klap)+Field(Ifi:Ilap,Jfim:Jlap,Kfim:Klap))
      Flux(Ifim:Ilap,Jfim:Jla,Kfim:Klap,2) = Flux(Ifim:Ilap,Jfim:Jla,Kfim:Klap,2) &
                                           + Vel(Ifim:Ilap,Jfim:Jla,Kfim:Klap,2) &
                                            *0.5*(Field(Ifim:Ilap,Jfim:Jla,Kfim:Klap)+Field(Ifim:Ilap,Jfi:Jlap,Kfim:Klap))
      Flux(Ifim:Ilap,Jfim:Jlap,Kfim:Kla,3) = Flux(Ifim:Ilap,Jfim:Jlap,Kfim:Kla,3) &
                                           + Vel(Ifim:Ilap,Jfim:Jlap,Kfim:Kla,3) &
                                            *0.5*(Field(Ifim:Ilap,Jfim:Jlap,Kfim:Kla)+Field(Ifim:Ilap,Jfim:Jlap,Kfi:Klap))
END SUBROUTINE FlxAddCDS !==================================================================
```

User Interaction - Visualization

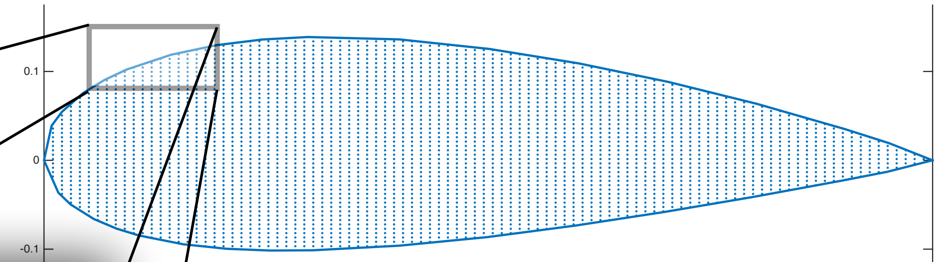


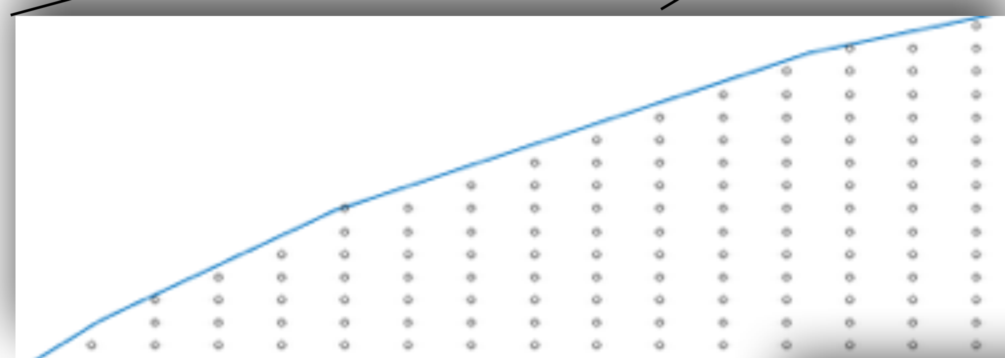
- Trigger events by keyboard, mouse, gamepad, smartphone
- The control-events are converted into the desired parameters
- Control parameters are sent to GPU-server
- Feedback from simulation are sent to the visualization computer via Xpra-Clone

Moving Boundary: Lagrange Particles



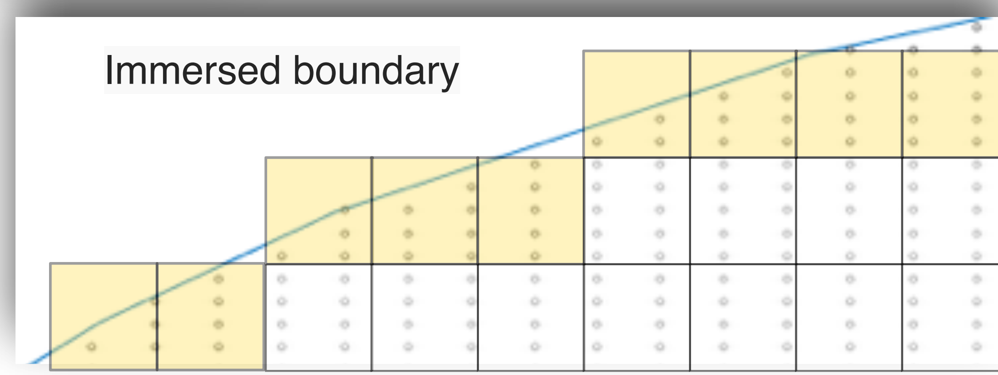
Clouds of Lagrangian particles



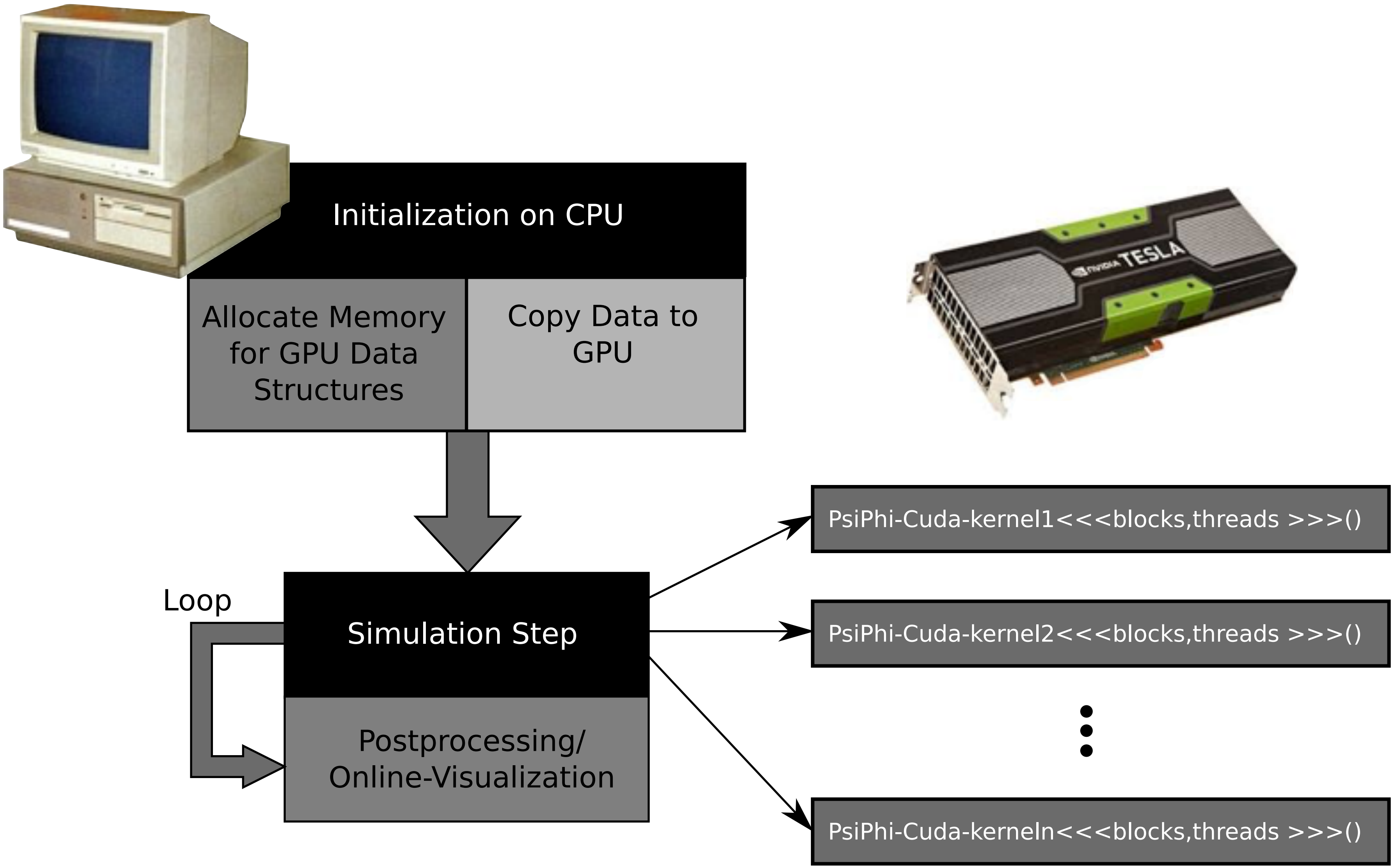


$$\frac{\partial p}{\partial x_i} = -\frac{\sum_{j=1}^{Nnb} \rho_j}{N_{nb}} a_{pi}$$

$$V_{si} = 2V_{pi} - \frac{\sum_{j=1}^{Nnb} V_{fij}}{N_{nb}}$$



Parallelization - GPU Solver

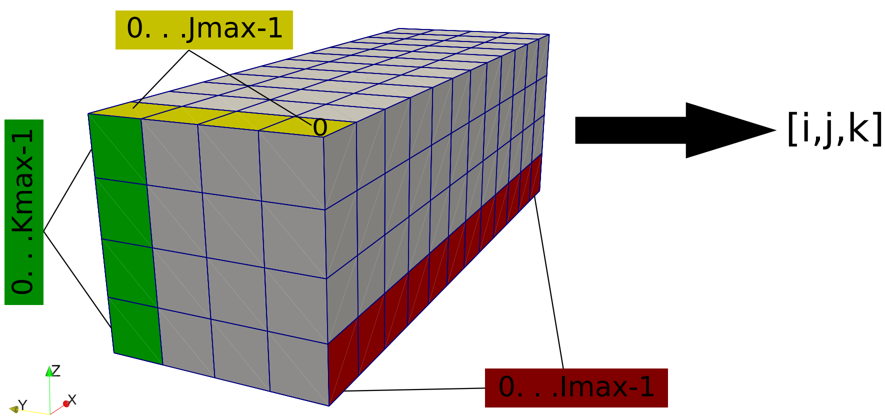


FORTRAN routines replaced by CUDA kernels

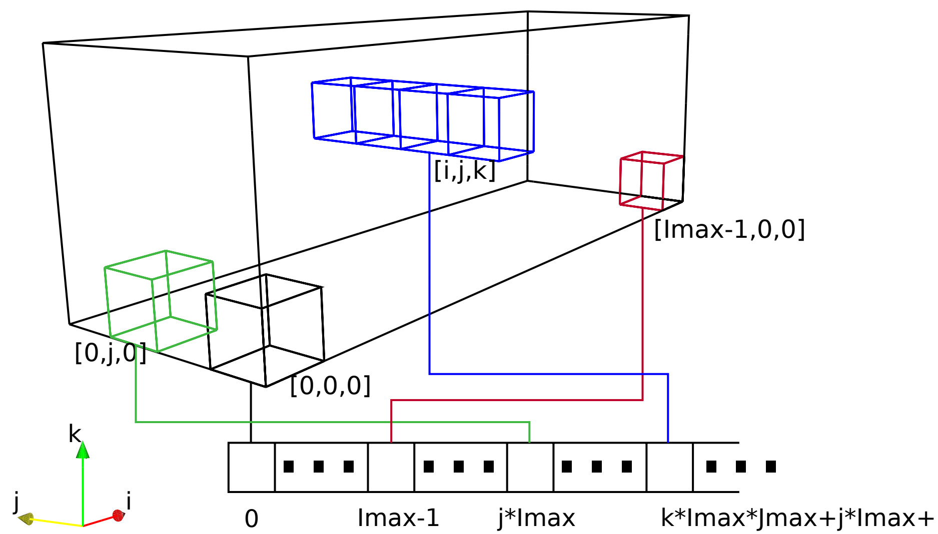
Mapping indices CPU vs. GPU

GPU support best for 1D arrays

Standard memory mapping of cell coordinates to 3D array index on CPU

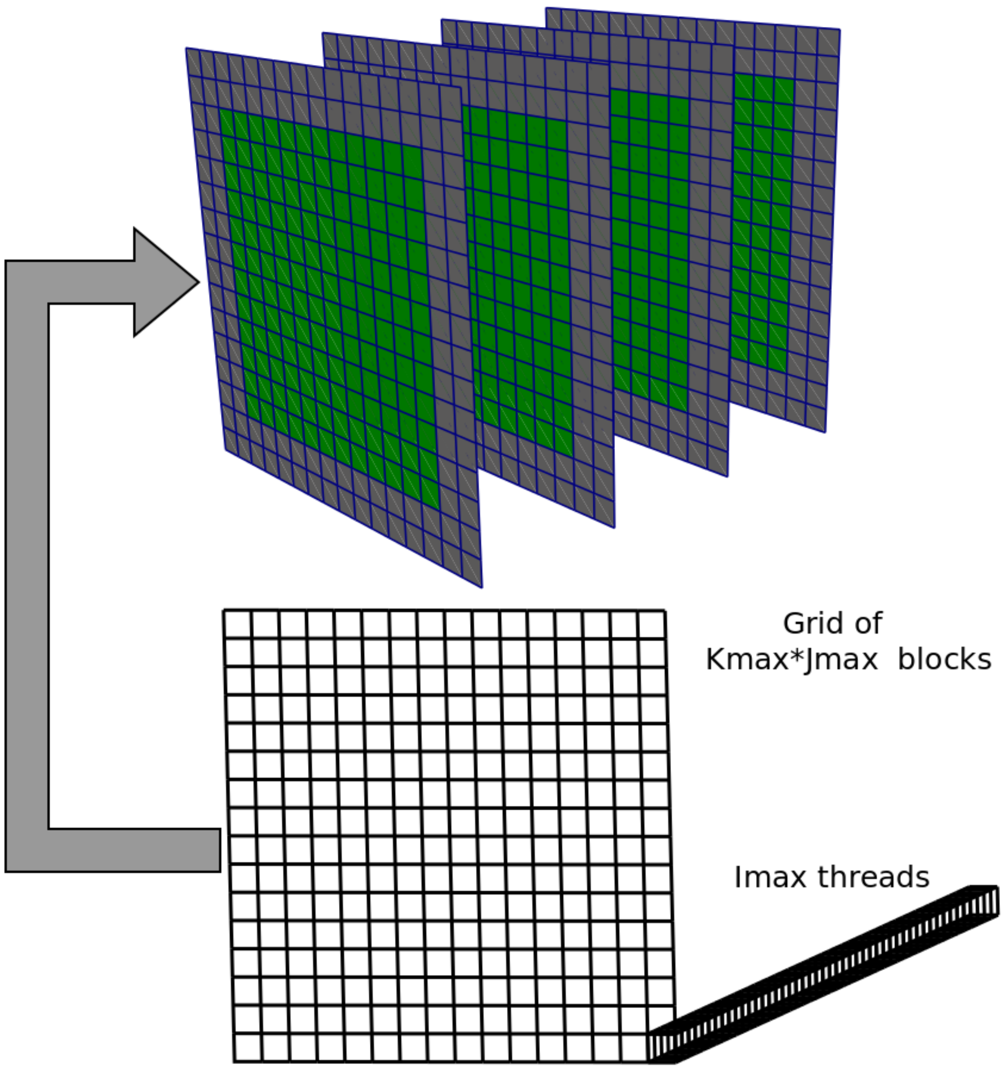


Mapping of cell coordinates to array index for 3D to 1D array mapping



$$(i,j,k) \mapsto k \cdot Imax \cdot Jmax + j \cdot Imax + i,$$
$$(i,j,k) \in [1,\dots,Imax] \times [1,\dots,Jmax] \times [1,\dots,Kmax]$$

GPU Kernels on an Cartesian Grid



PsiPhi-Cuda-kernel1<<<blocks,threads >>>()

PsiPhi-Cuda-kernel2<<<blocks,threads >>>()

⋮

PsiPhi-Cuda-kerneln<<<blocks,threads >>>()

Coupling between CPU and GPU

technische universität dortmund

UNIVERSITÄT DUISBURG ESSEN

CPU

1. Generate immersed boundary
2. Initialize flow field

3. Copy data from CPU to GPU

Loop

4.

4. Call GPU solver

5. Copy data from GPU to CPU

GPU

1. Allocate the GPU arrays
2. Initialize all the GPU arrays

3. Receive data from CPU

4. Solve transport equations

5. Compute pressure gradient & update momentum

6. Compute inlet & outlet boundary conditions using parameters from devices

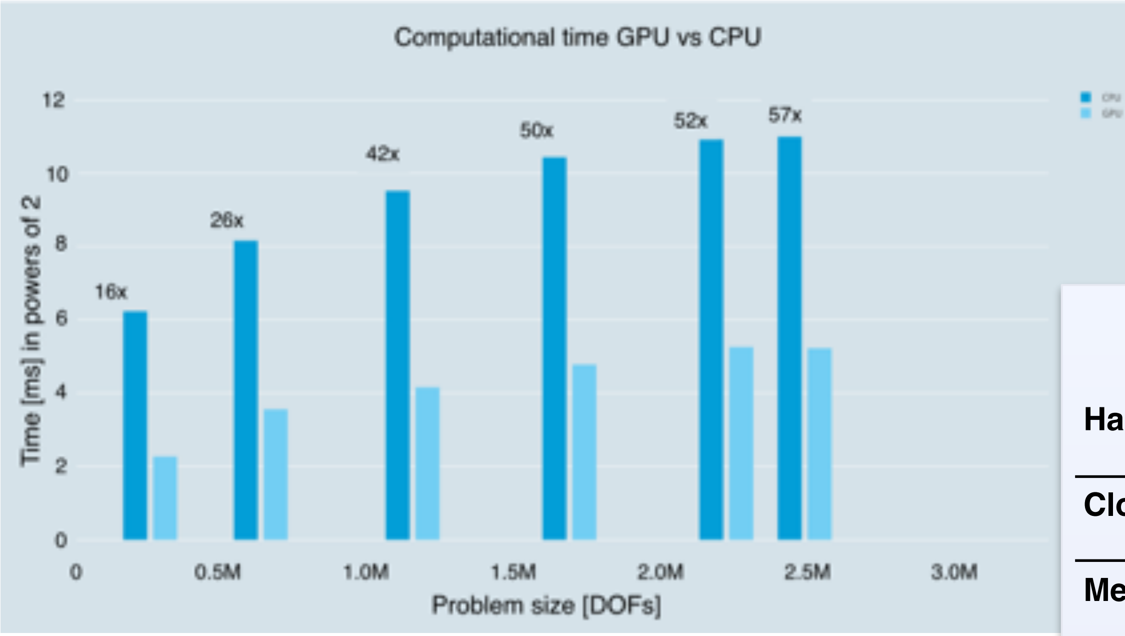
7. Compute the moving boundaries using parameters from devices

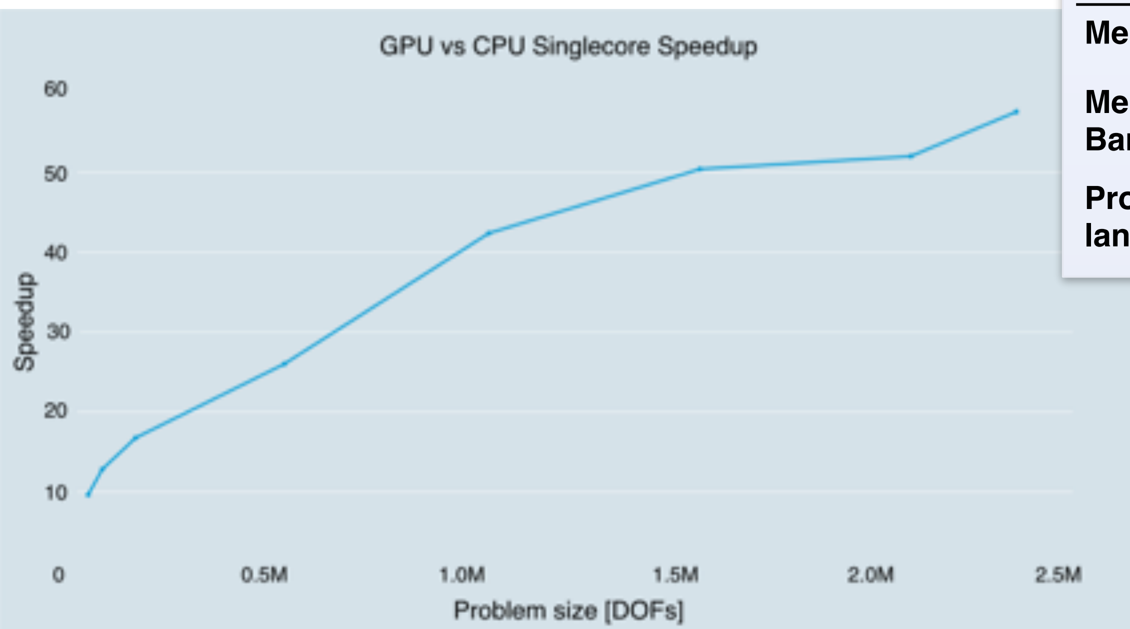
8. Compute sub-grid turbulence

9. Visualize the data using parameters from devices

10. Copy back data to CPU

Benchmark GPU vs CPU

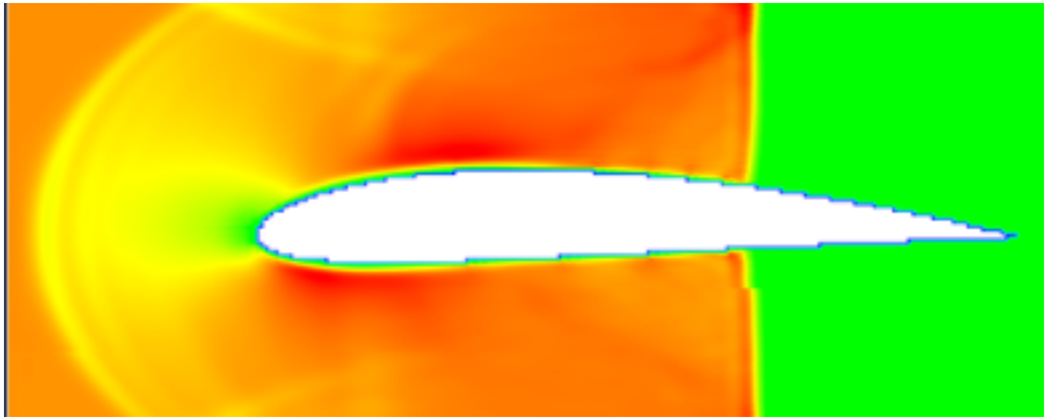




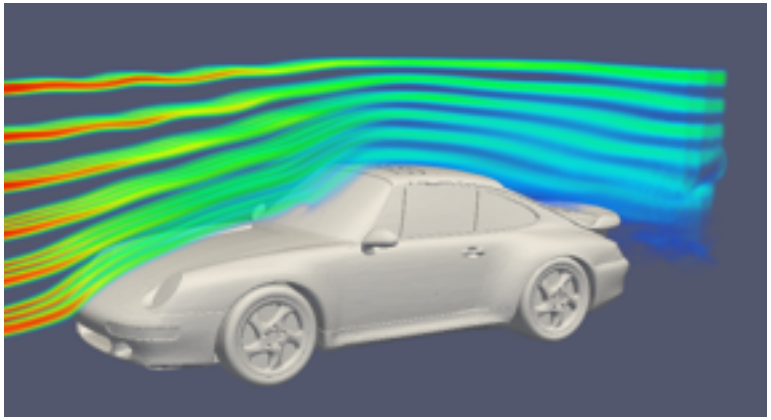
	GPU	CPU
Hardware	NVIDA Tesla K20Xm	Intel Xeon E5-260
Clock speed	732 MHz	3.3 GHz
Memory clock	1300 MHz	1600 MHz
Memory size	6GB (GDDR5)	64 GB
Memory Bandwidth	250GB/s	
Programming languages	C CUDA	Fortran 90

Some Test Cases by Wind Tunnel

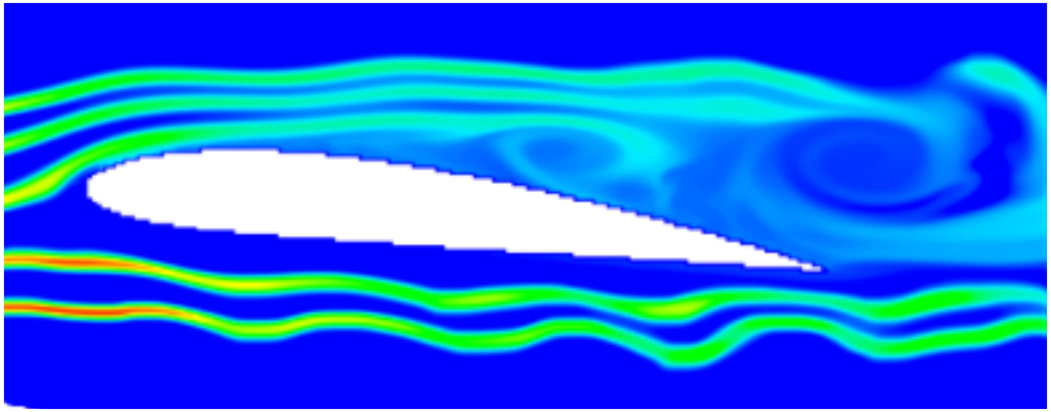
Pressure wave (Airfoil NACA 4415)



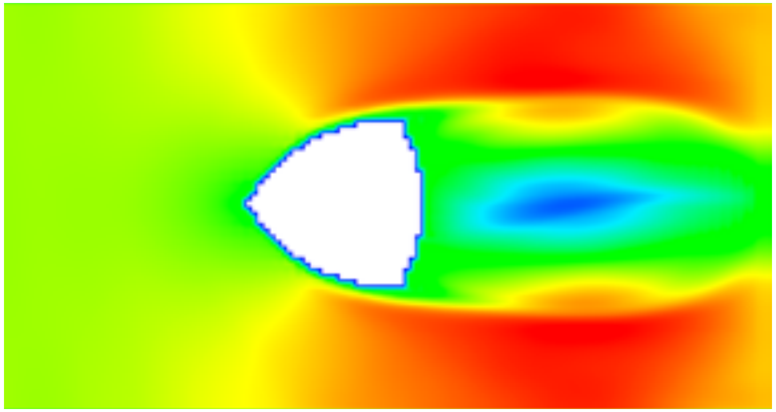
Passive tracer in 3D around a car



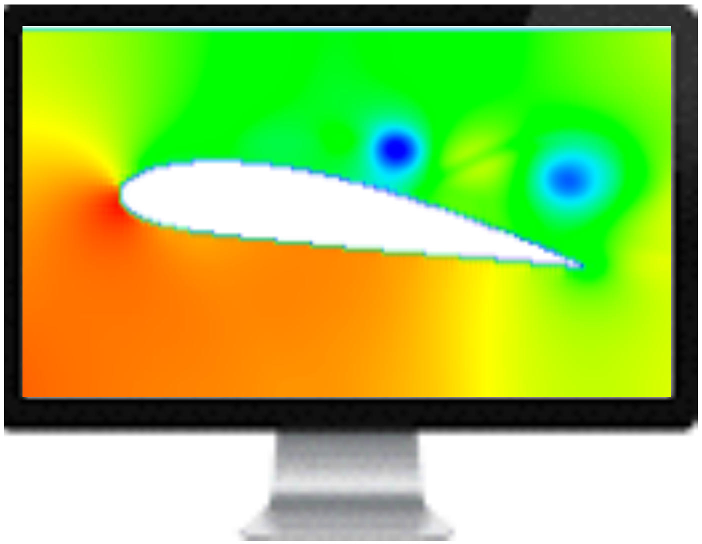
Passive tracer (Airfoil NACA 4415)

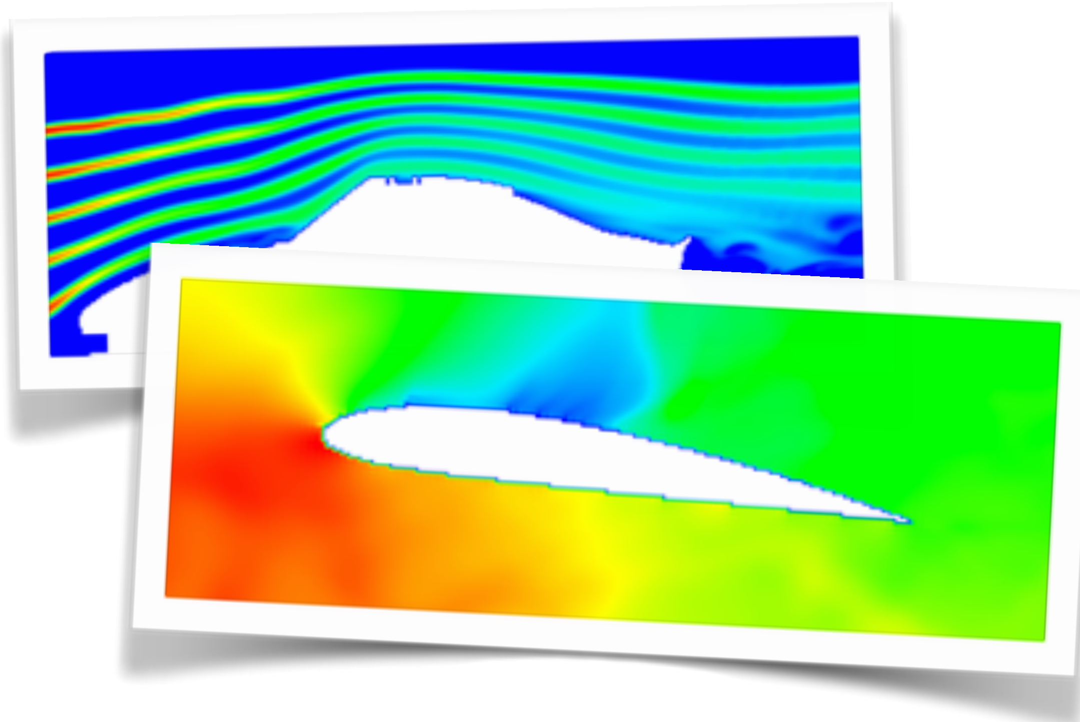


Velocities field around a Reuleaux triangle



VINKanal Provides

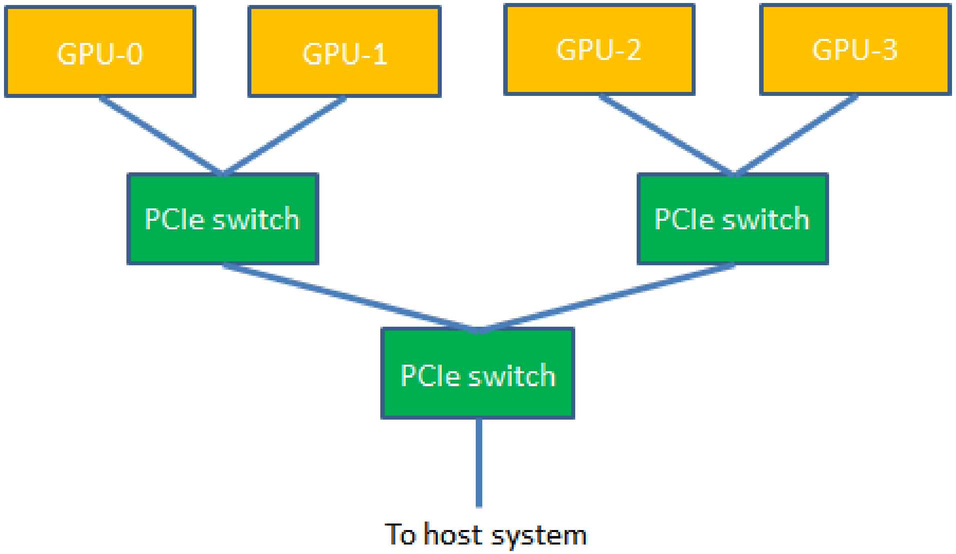




1. **Quick access to the aerodynamics of complex geometries**
2. **„Wind tunnel tests“ can be performed or prepared on small computers**
3. **Simulated quantities are visualized instantly during the runtime**
4. **User can interact with the simulation directly**
5. **Transient processes can be studied effectively**

But: Reynolds number is still the major challenge

Future Work



1. Multi CPU & Multi GPU
2. Incompressible solver
3. Local refinement
4. Include more scenarios f
5. Improved In-situ Visualization
6. Improved architecture
7. Non-visual output and evaluation
8. Better turbulence modeling
9. Packaging for engineering, research and education

Conclusions

- A fast, efficient solver was implemented on GPUs
- Complex geometries are handled in a simple manner
- Real time interaction is integrated
- Data visualization is available

technische universität dortmund

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

CCSS

Thank you for your attention!

We gratefully acknowledge funding by the Mercator Foundation through a MERCUR project.

Mercator Research Center Ruhr

Eine Initiative der Stiftung Mercator und der Universitätsallianz Ruhr