A Scalable GPU-Based Compressible Fluid Flow Solver for Unstructured Grids

Patrice Castonguay and Antony Jameson

Aerospace Computing Lab, Stanford University

GTC Asia, Beijing, China

December 15th, 2011

Antony Jameson

- Revolutionized CFD in aeronautics

  Solution to full potential equation, efficient multi-grid methods, shock capturing for transonic flows, control theory for shape optimization

- Lead developer of FLO and SYN codes used throughout the aerospace industry

- Over 400 scientific papers

- Multiple honorary awards

- Trademark: Fast codes

SD++

- 2D/3D compressible viscous flow solver
- Mixed grids of quadrilaterals and triangles in 2D and hexahedra, prisms and tetrahedra in 3D
- Arbitrary order of accuracy
- Solver can run on multiple CPUs or GPUs (C++/Cuda/MPI)

Talk Overview

- Part 1: Unstructured High-Order Methods
  - Why are they useful?
- Part 2: Flux Reconstruction Method for the Navier-Stokes equations
  - Algorithm details
  - Why it's a good fit for GPUs
- Part 3: GPU Implementation Details
  - Single-GPU: Efficient use of GPU memory hierarchy
  - Multi-GPU : How to obtain good scalability
- Part 4: Performance analysis and Applications
  - Performance on a single GPU
  - Strong and weak scaling study
  - How GPUs enable previously intractable fluid flow simulations

Unstructured High-Order Methods

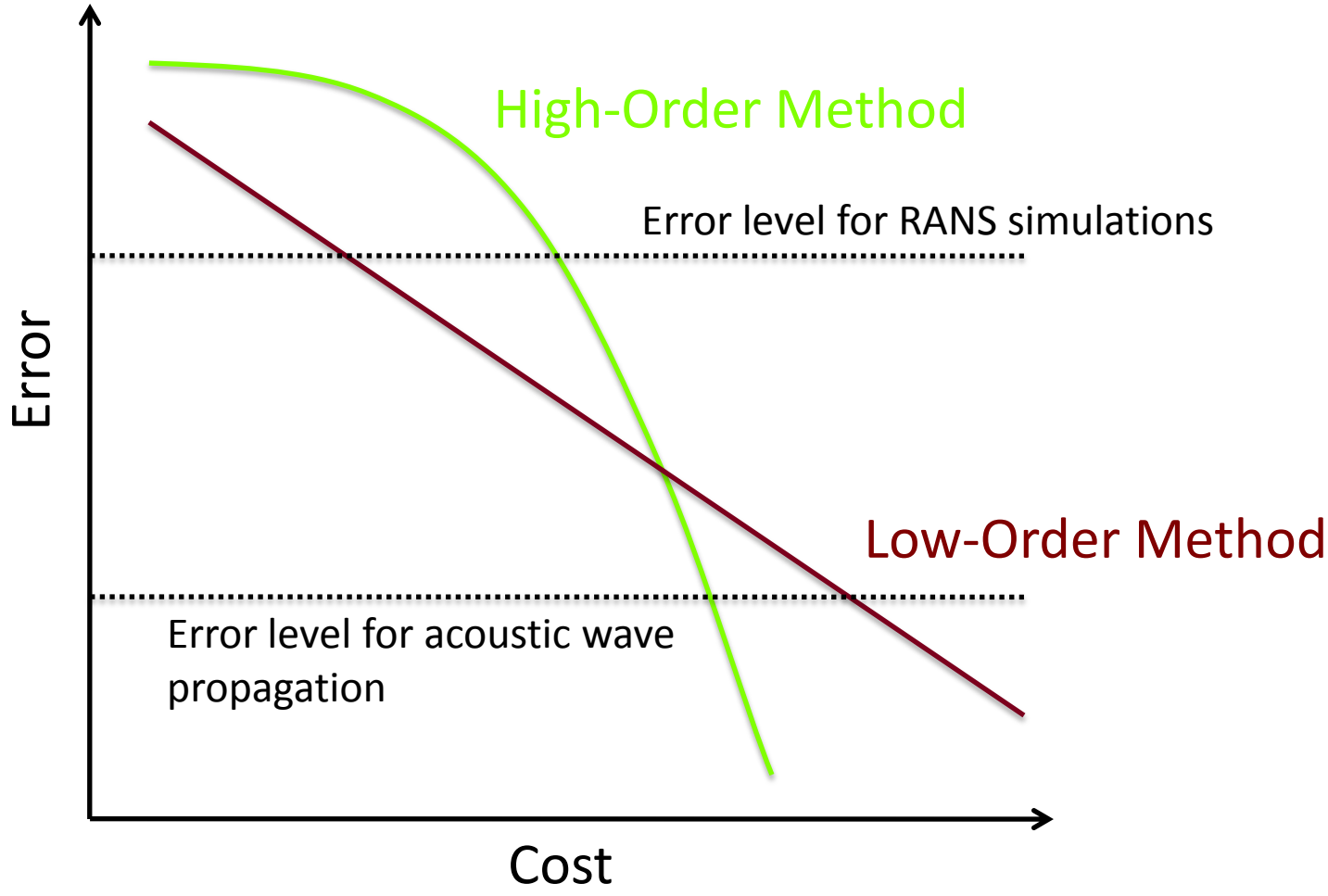
- What does high-order mean?

- Low-order methods:
  - Order of accuracy is 1 or 2 (Error is of order h or order h^2)
  - Robust and simple to implement
  - Dissipative

- High-order methods:
  - Order of accuracy is > 2
  - Not as mature as low-order methods
  - More work per DOF
  - Required for applications where accuracy requirement is high

Unstructured High-Order Methods

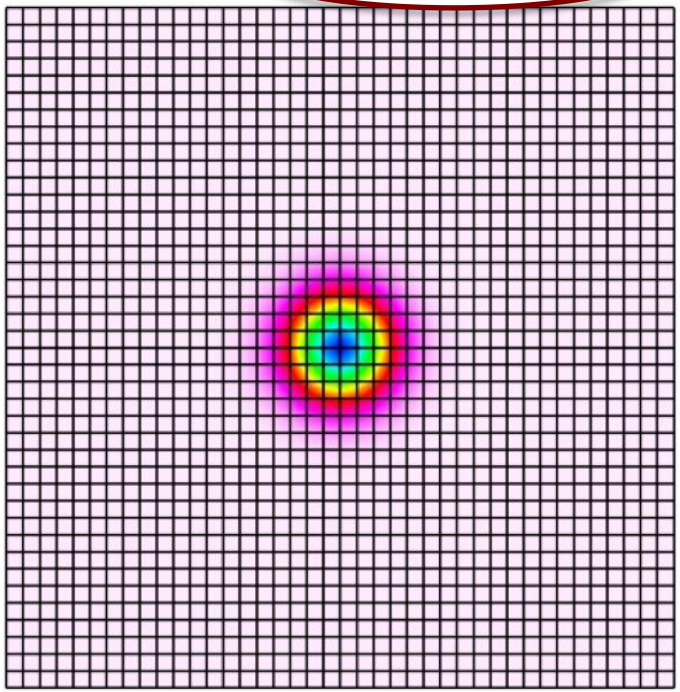
- Why do we need high-order methods?



Unstructured High-Order Methods

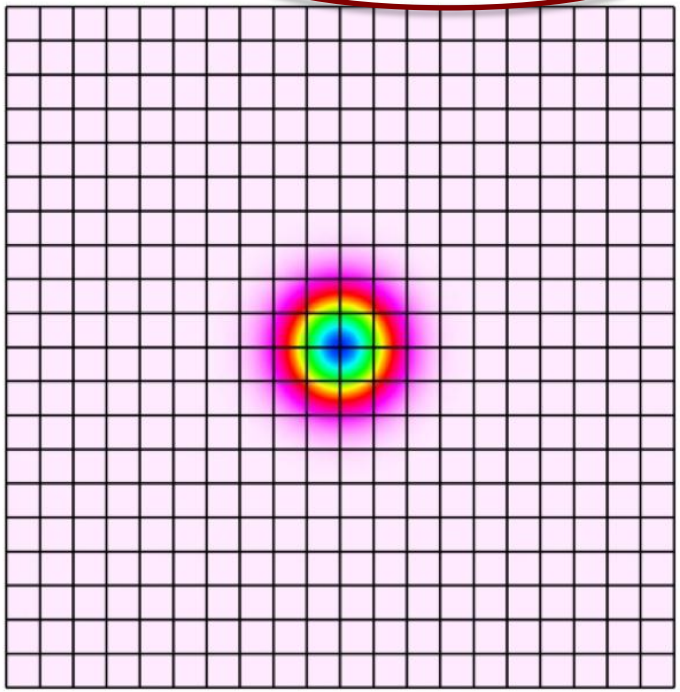
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 0

4th order (25,600 DOFs)

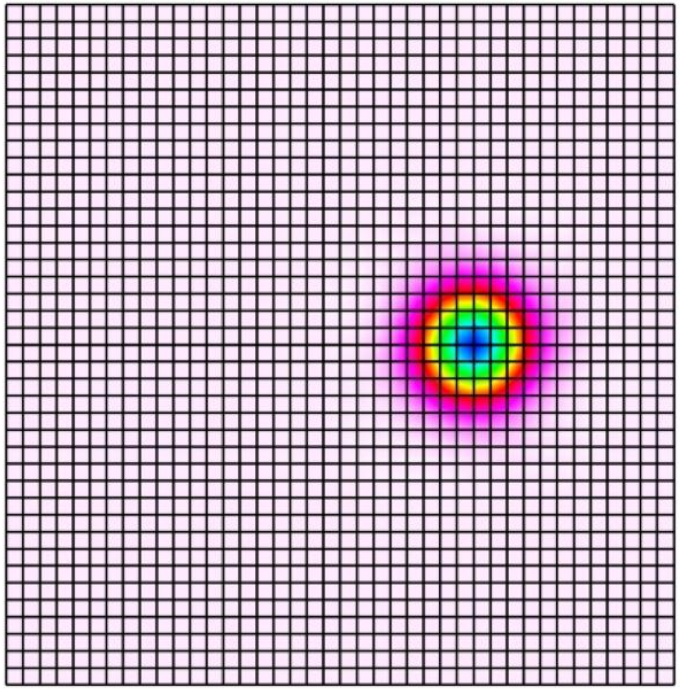


t = 0

Unstructured High-Order Methods

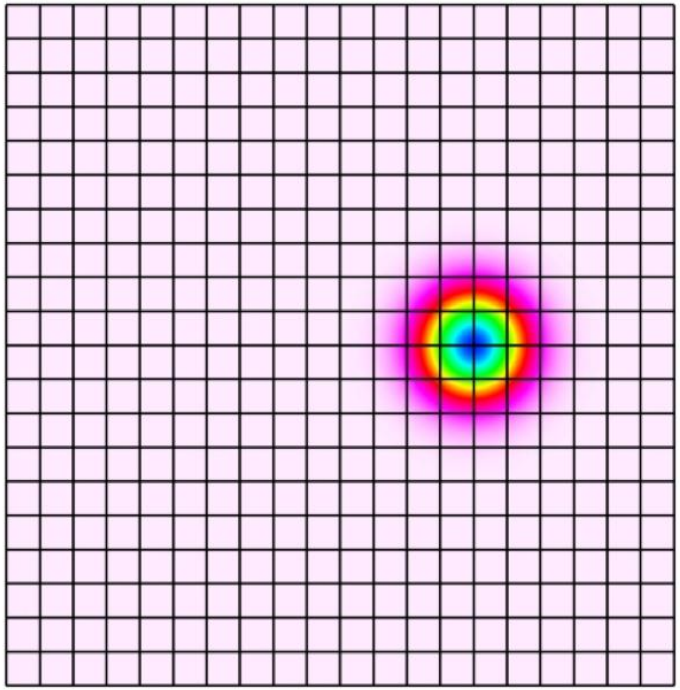
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 1

4th order (25,600 DOFs)

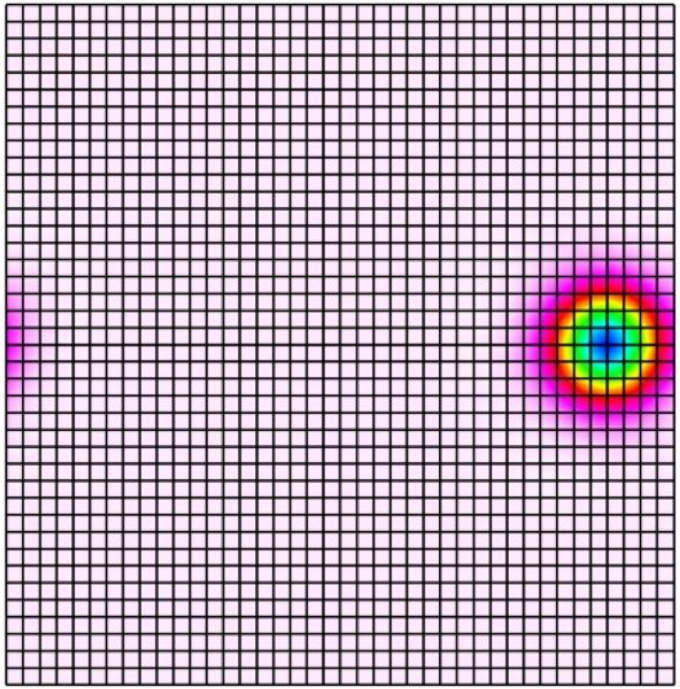


t = 1

Unstructured High-Order Methods

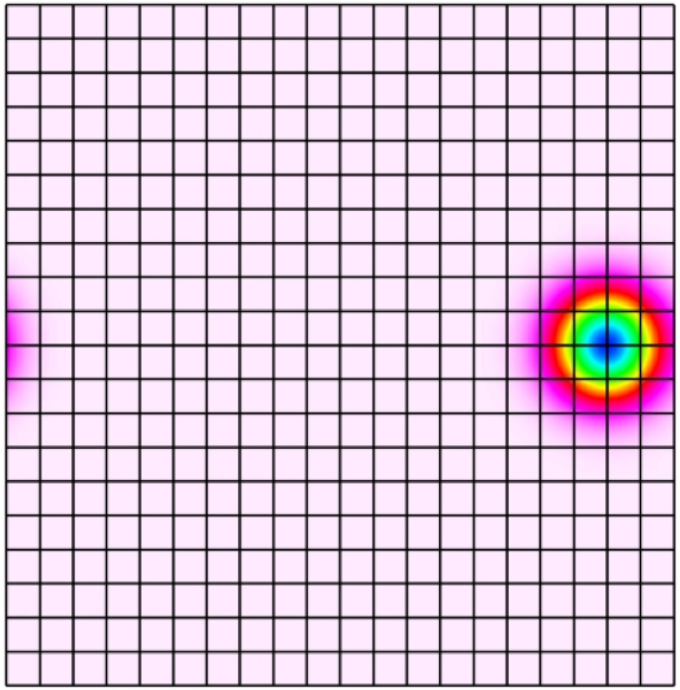
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 2

4th order (25,600 DOFs)

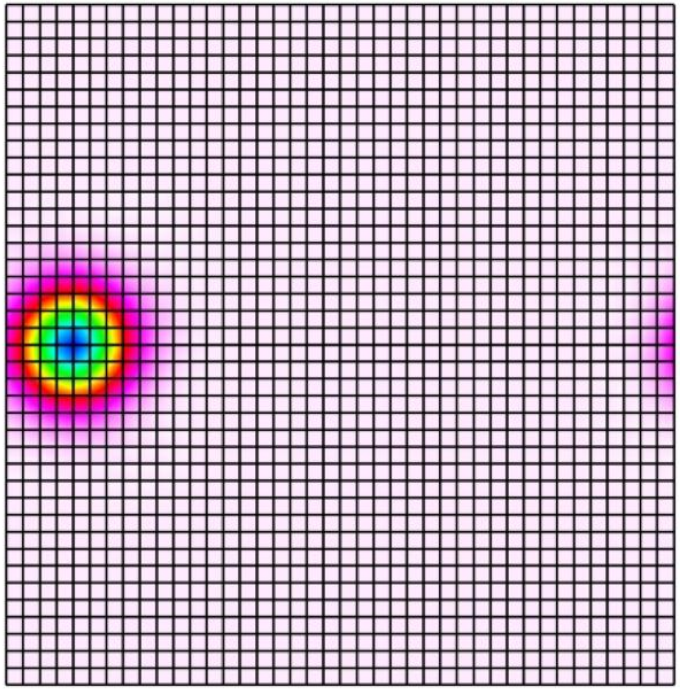


t = 2

Unstructured High-Order Methods

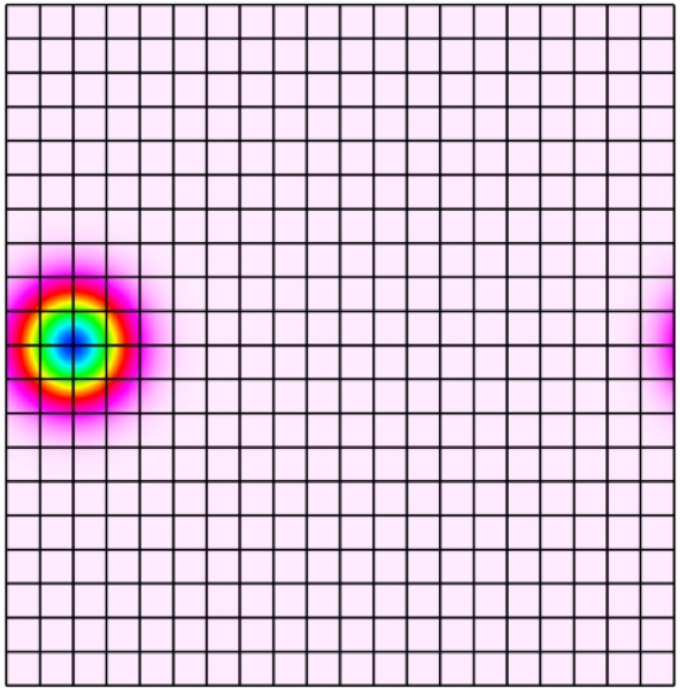
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 3

4th order (25,600 DOFs)

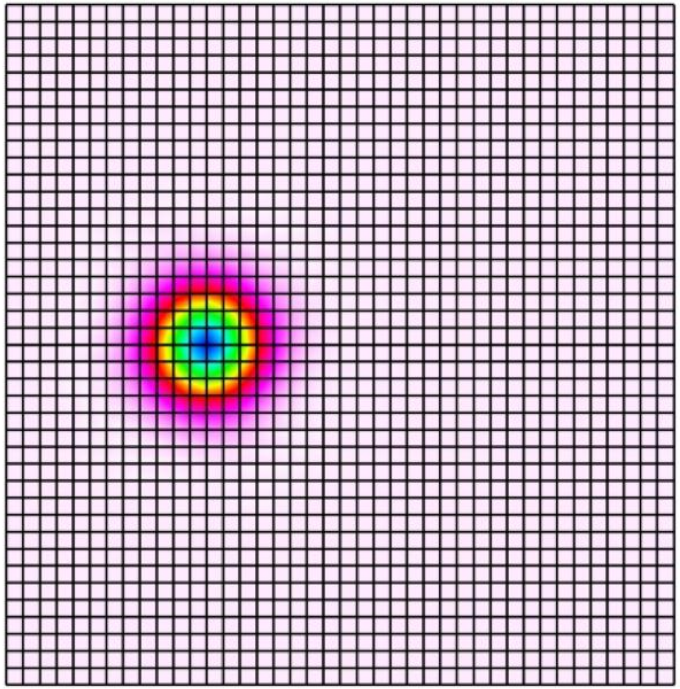


t = 3

Unstructured High-Order Methods

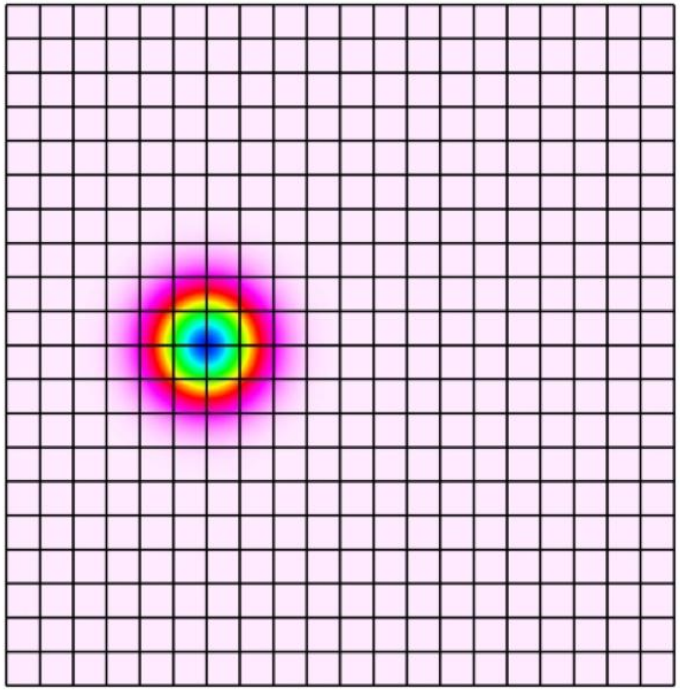
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 4

4th order (25,600 DOFs)

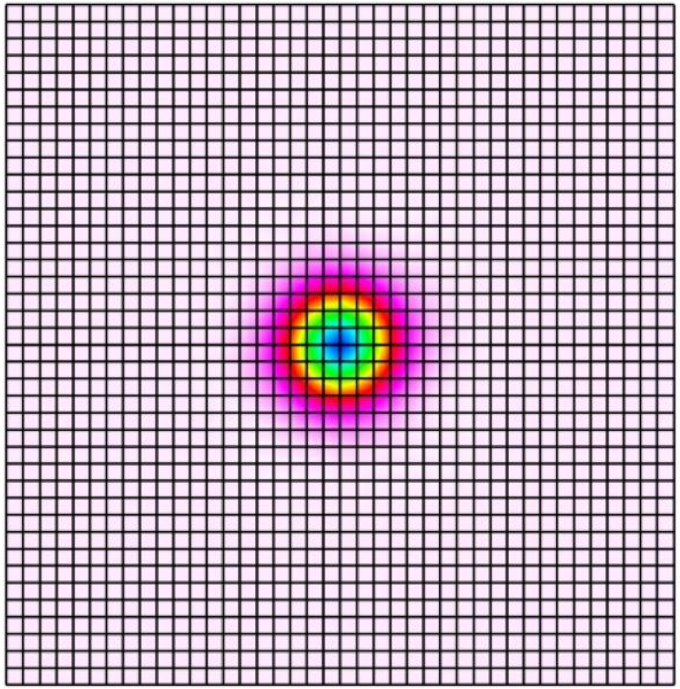


t = 4

Unstructured High-Order Methods

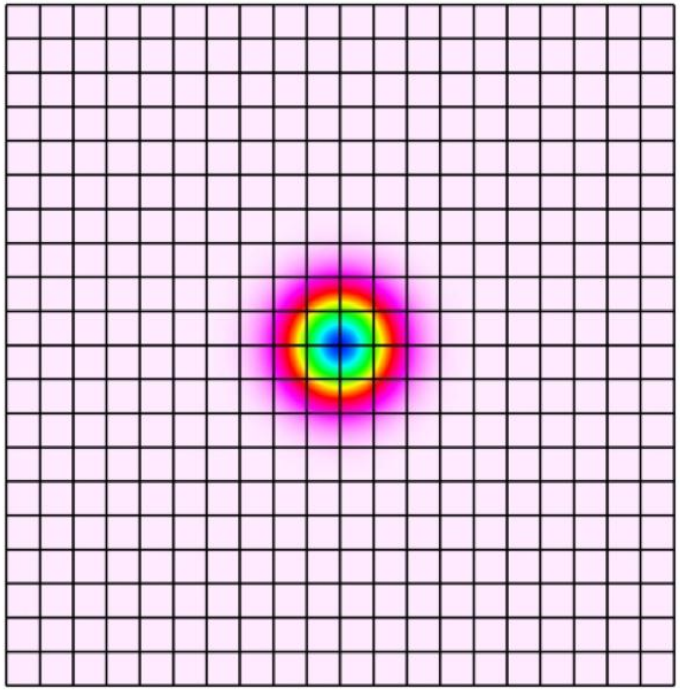
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 5

4th order (25,600 DOFs)

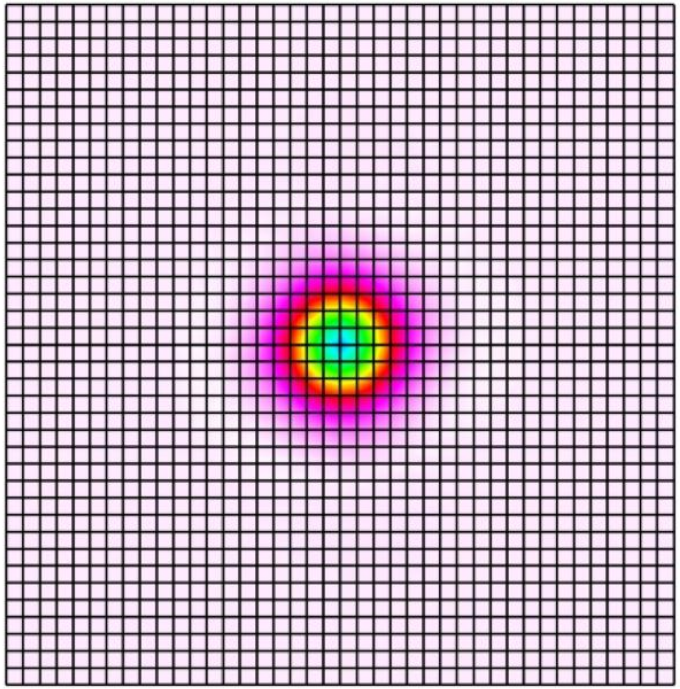


t = 5

Unstructured High-Order Methods

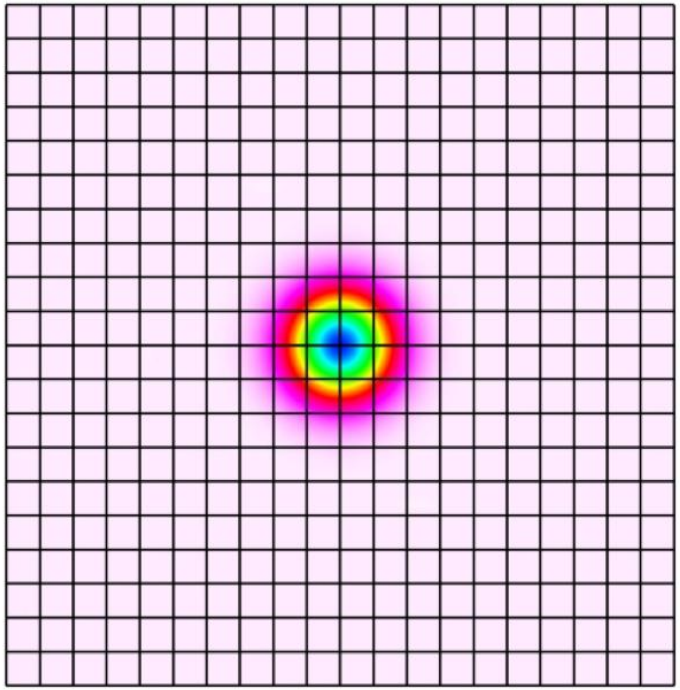
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 20

4th order (25,600 DOFs)

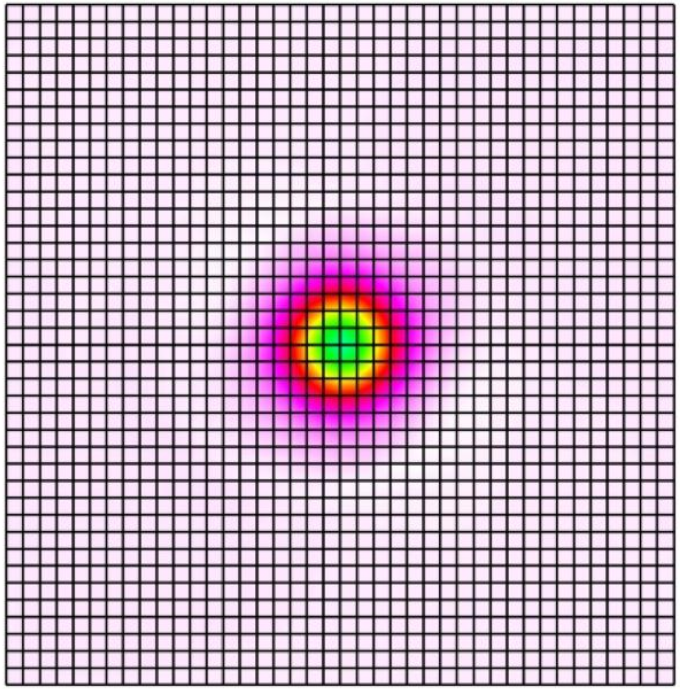


t = 20

Unstructured High-Order Methods

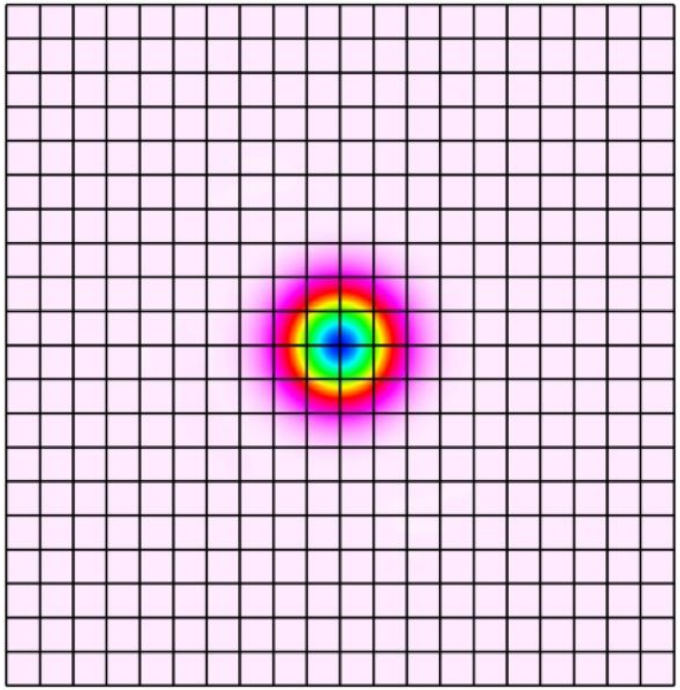
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 40

4th order (25,600 DOFs)

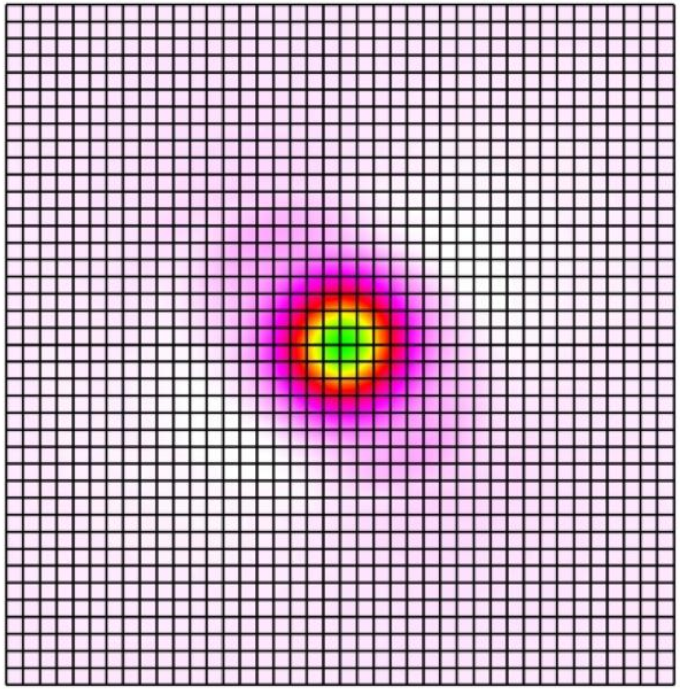


t = 40

Unstructured High-Order Methods

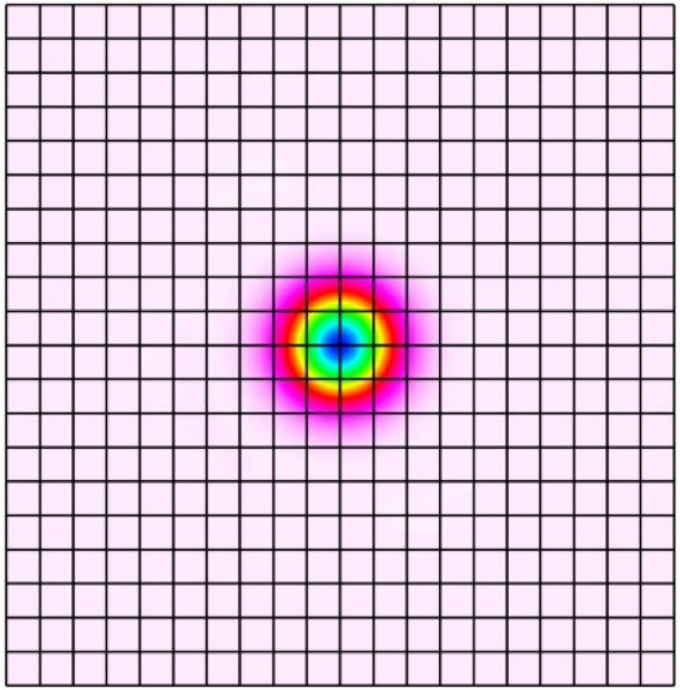
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 60

4th order (25,600 DOFs)

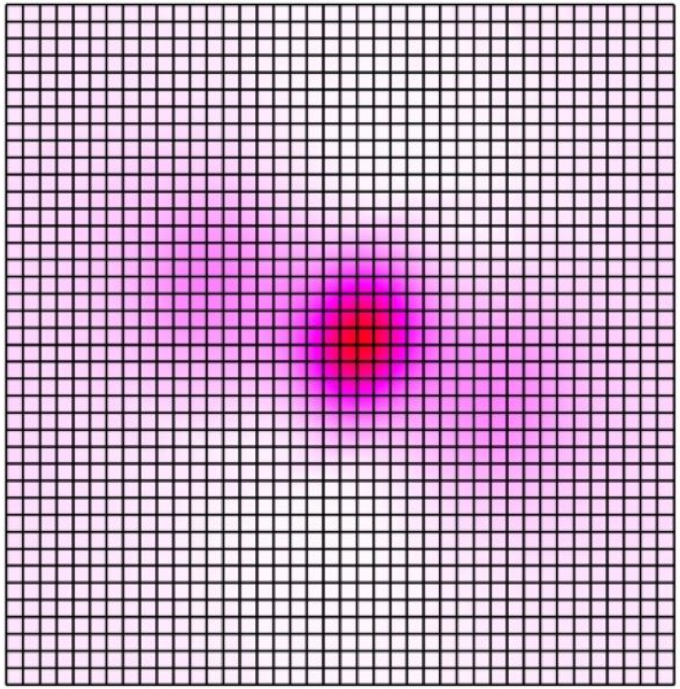


t = 60

Unstructured High-Order Methods

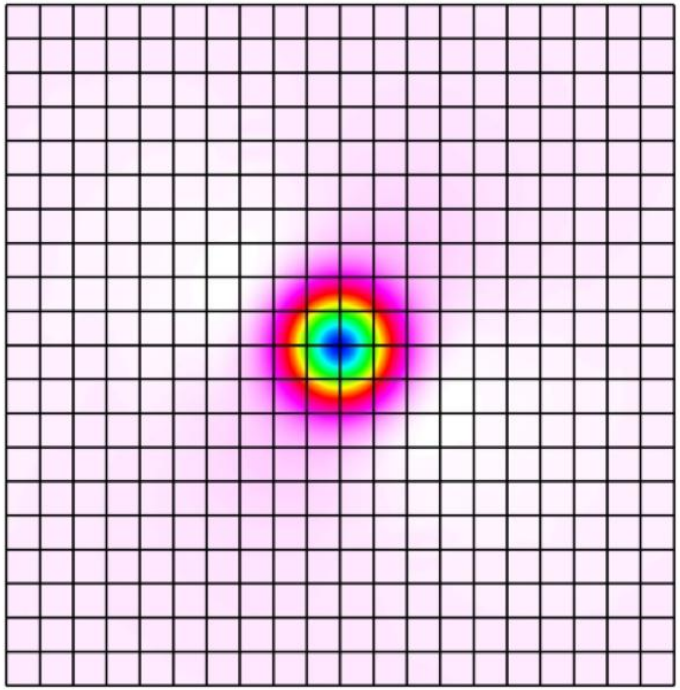
- Why is high-order useful?

2nd order (25,600 DOFs)



t = 180

4th order (25,600 DOFs)



t = 180

Unstructured High-Order Methods

- Why are they useful:

Complex geometry + High Accuracy

- In computational fluid dynamics, they enable:
  - Simulation of wave propagation over long distances in vicinity of complex geometries
  - Simulation of vortex motion over long distances in vicinity of complex geometries
  - Effective Large Eddy Simulations (LES) in vicinity of complex geometries

Unstructured High-Order Methods

- Airframe noise (turbulence + generation/propagation of sound waves + complex geometry)

Unstructured High-Order Methods

- Rotorcraft (turbulence + track vortices over long distances + complex geometry)

Unstructured High-Order Methods

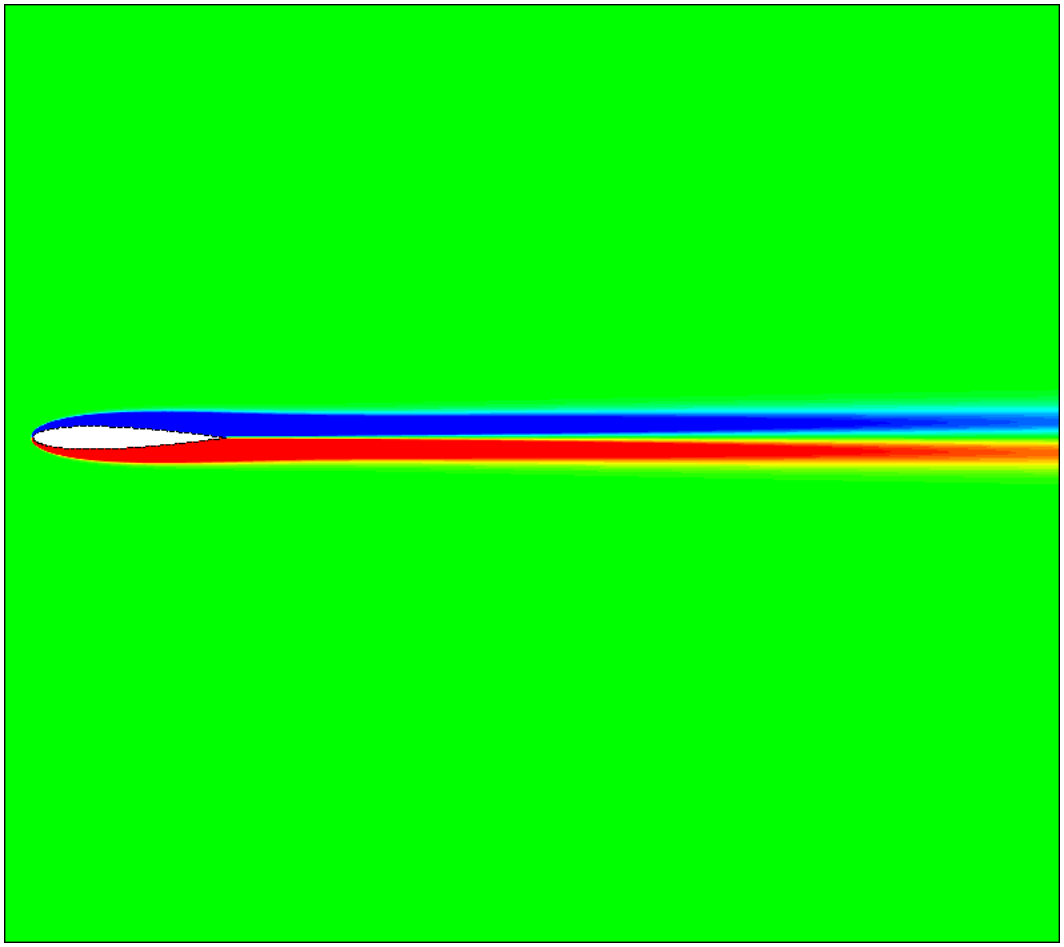
- Flapping wing flight (transitional Reynolds number + vortex dominated flow + complex geometry)

Unstructured High-Order Methods

- Flapping wing flight (transitional Reynolds number + vortex dominated flow + complex geometry)

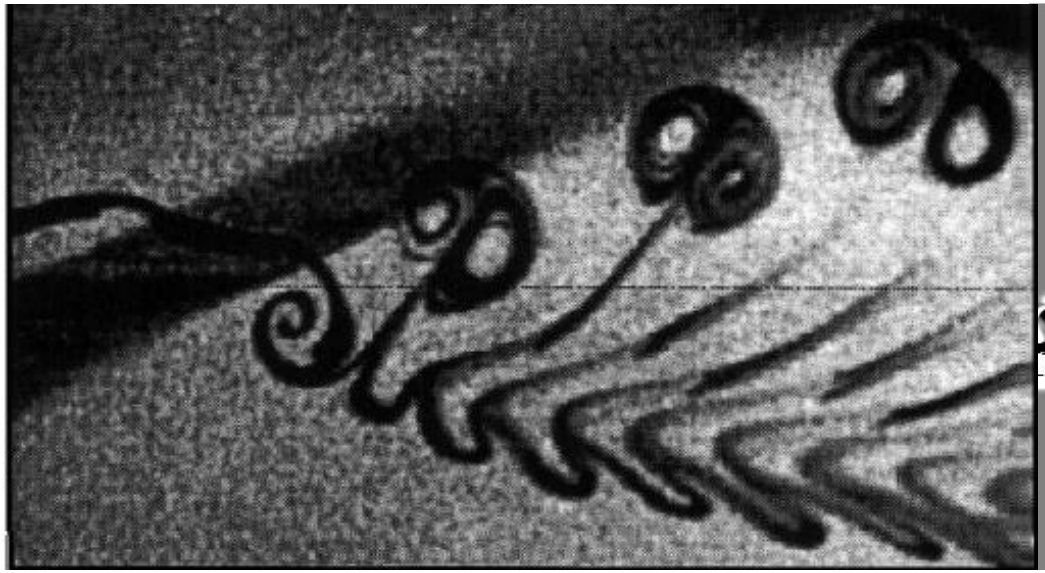
Unstructured High-Order Methods

- Plunging airfoil: zero AOA, Re=1850, Frequency: 2.46 rad/s
- 5^{th} order accuracy in space, 4^{th} order accurate RK time stepping

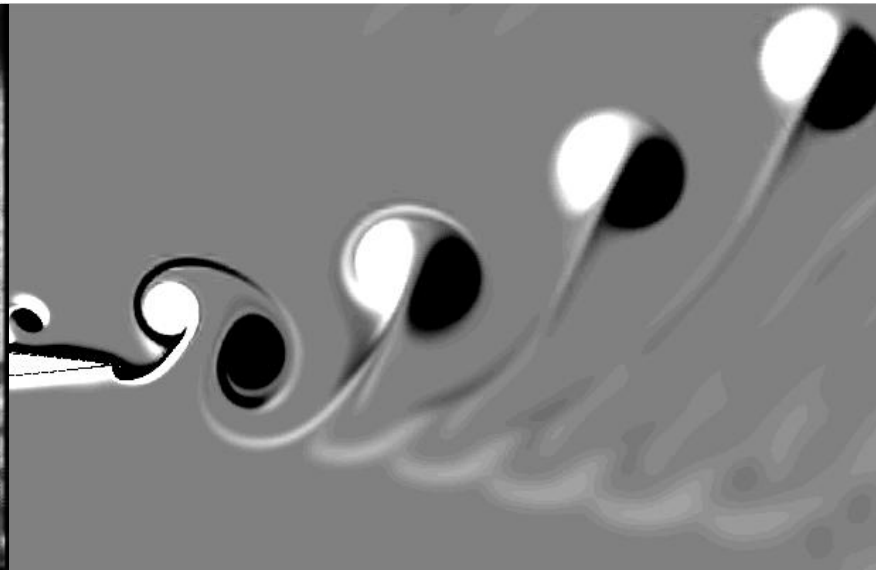


Unstructured High-Order Methods

- Vortical patterns and force coefficients agree with experiments
- Able to capture the fine structures in addition to main vortex train



Experiment by Jones, Dohring, Platzer, July 1998



Vorticity contours, 5^{th} order accuracy solution

Unstructured High-Order Methods

- Computations are demanding:
  - Millions of DOFS
  - Hundreds of thousands of time steps
- Until recently, high-order simulations over complex 3D geometries were intractable, unless you had access to large cluster
- GPUs to the rescue!

Flux Reconstruction Method

- For a conservation law in strong form

$$\frac{\partial Q}{\partial t} + \nabla \cdot \mathbf{F}(Q, \nabla Q) = 0$$

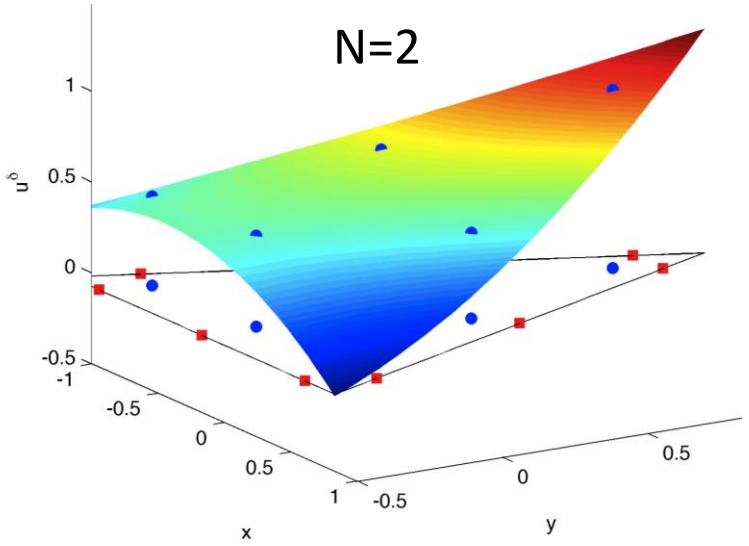
- Ex: Euler equations

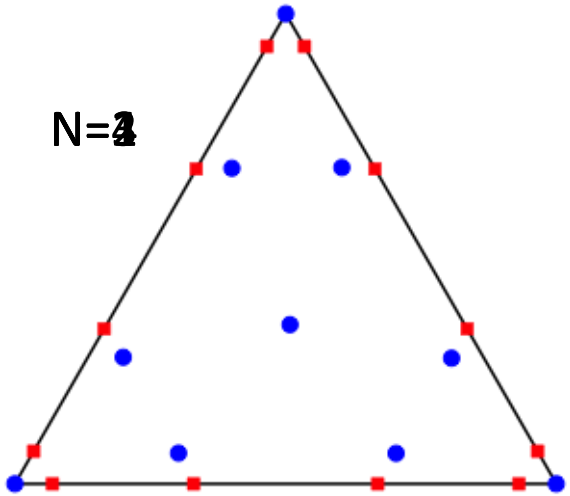
$$\mathrm{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \mathrm{F}_x = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho ue + p \end{bmatrix}, F_y = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho ve + p \end{bmatrix}$$

- Solve differential form within each element, with boundary data from neighbouring elements
- Can recover Spectral Difference and Discontinuous Galerkin methods

Flux Reconstruction Method

- Solution in each element approximated by a multi-dimensional polynomial of order N
- Order of accuracy: h^{N+1}
- Multiple DOFs per element



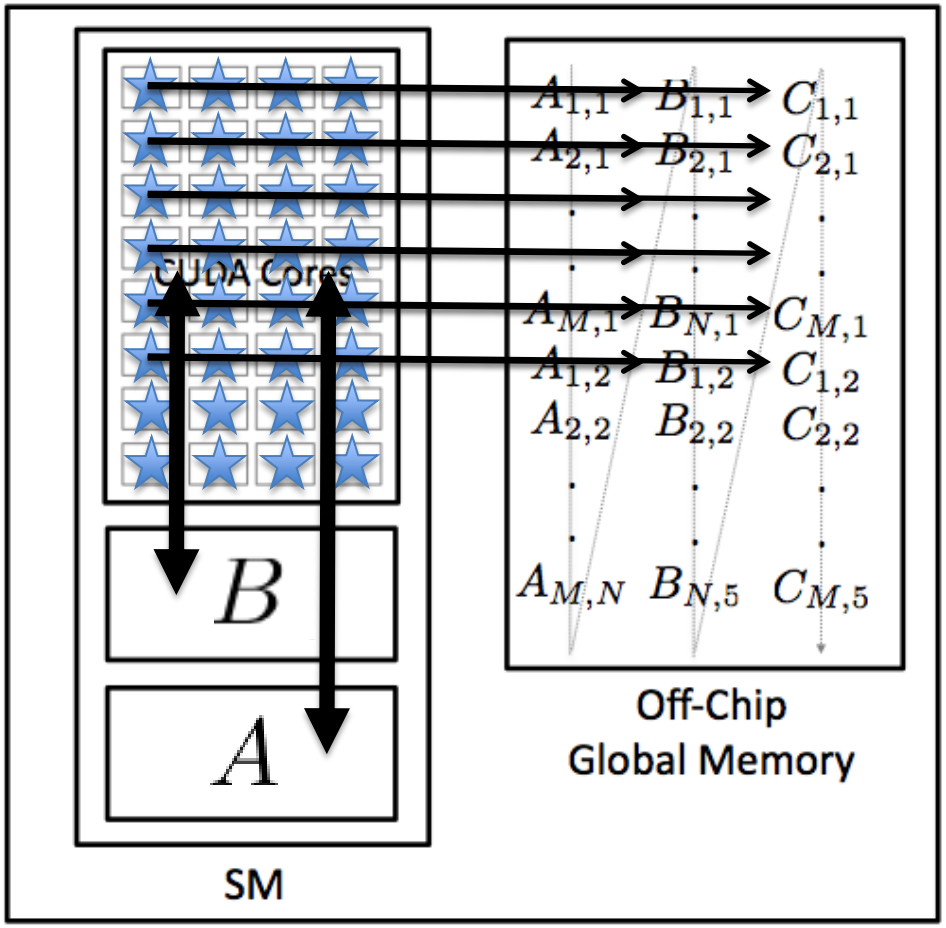


Flux Reconstruction Method

- Method maps well to the GPUs:
  - High-level of parallelism (millions of DOFs)
  - More work per DOF compared to low-order methods (flops are “free” on a GPU)
  - Cell-local operations benefit from fast on-chip shared-memory

GPU Implementation

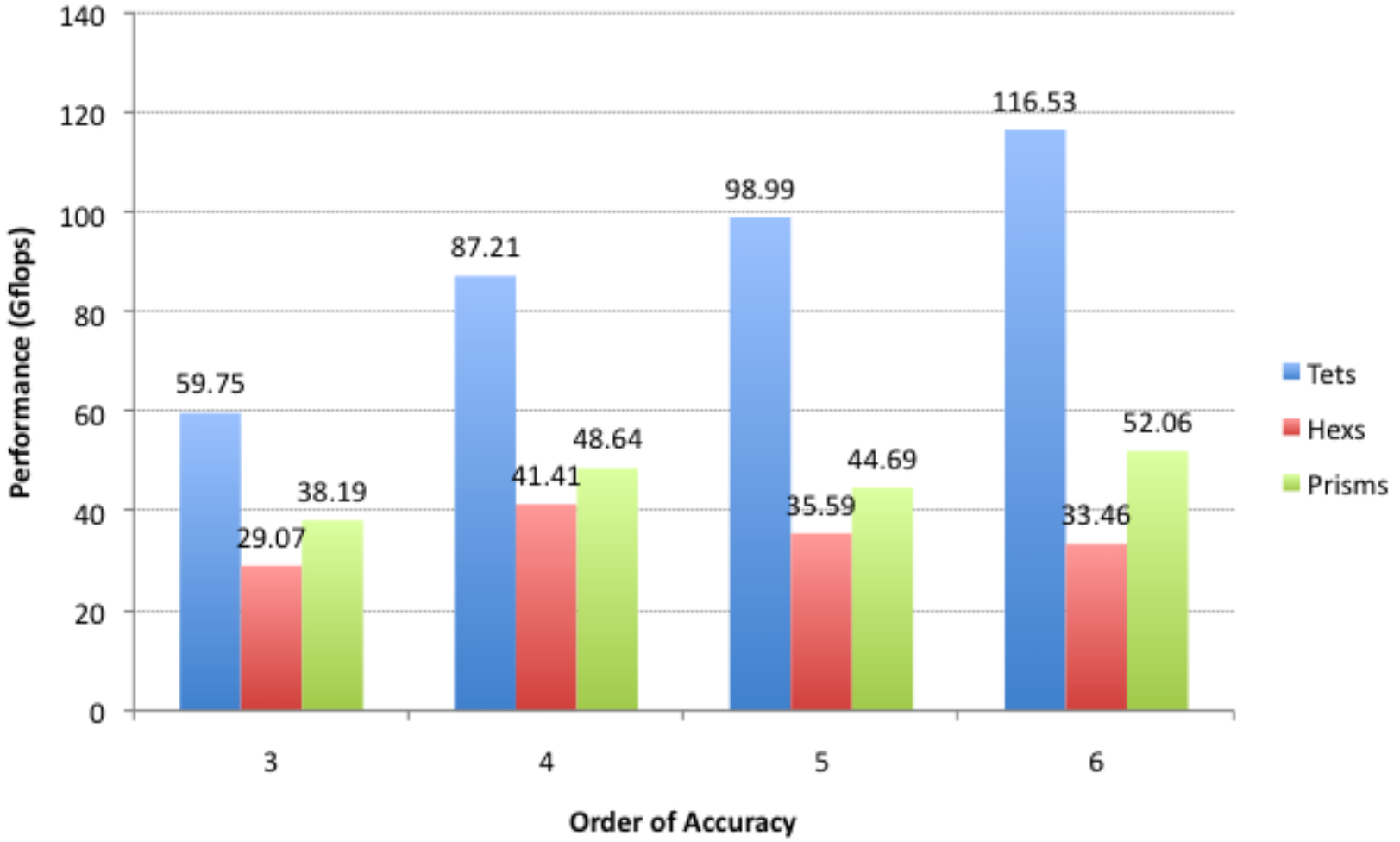
$$\begin{bmatrix} A_{1,1} & . & . & A_{1,N} \\ A_{2,1} & . & . & A_{2,N} \\ . & . & . & . \\ A_{M,1} & . & . & A_{M,N} \end{bmatrix} \begin{bmatrix} B_{1,1} & . & . & B_{1,5} \\ B_{2,1} & . & . & B_{2,5} \\ . & . & . & . \\ B_{N,1} & . & . & B_{N,5} \end{bmatrix} = \begin{bmatrix} C_{1,1} & . & . & C_{1,5} \\ C_{2,1} & . & . & C_{2,5} \\ . & . & . & . \\ C_{M,1} & . & . & C_{M,5} \end{bmatrix}$$



GPU Implementation

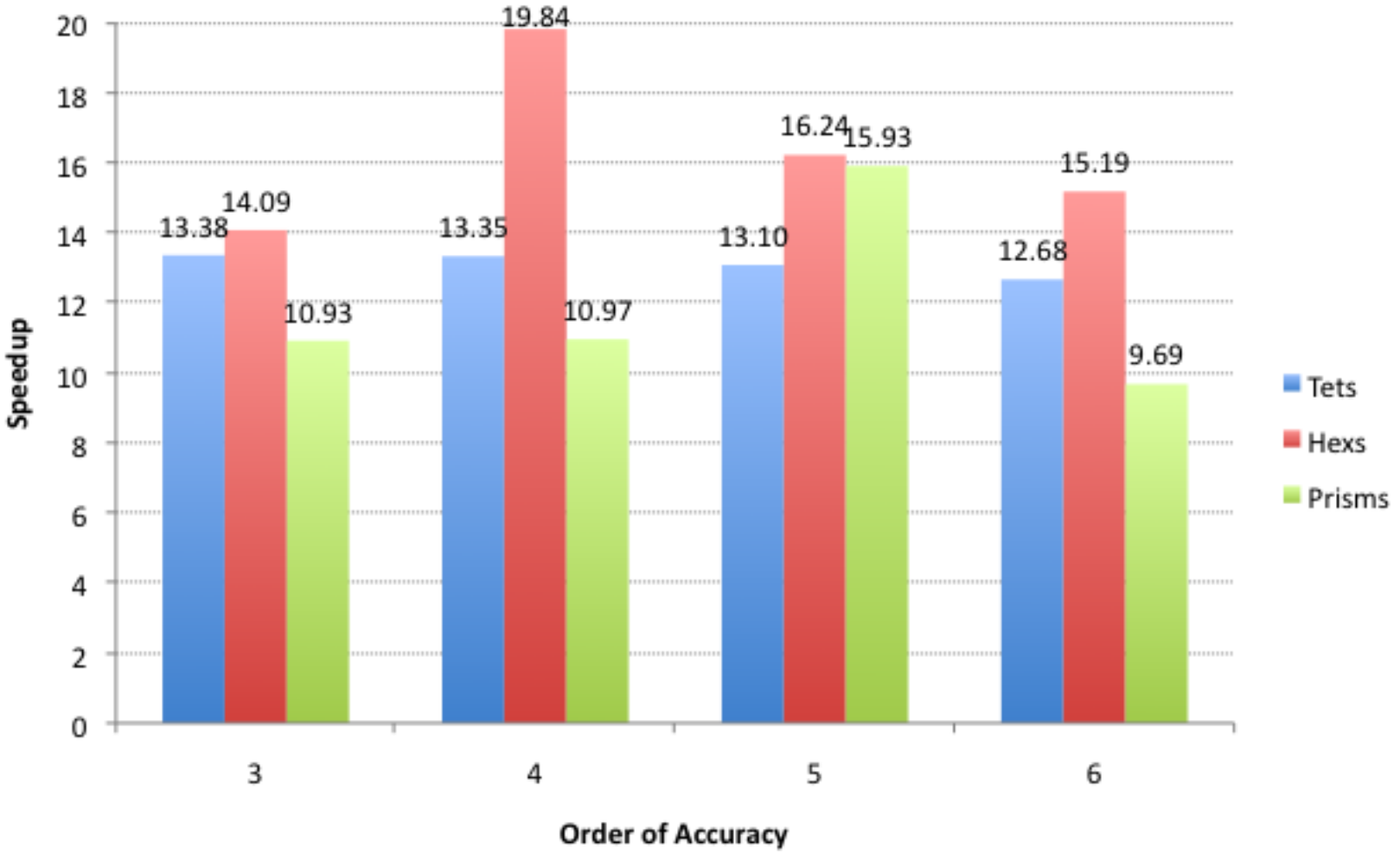
- Test case: Viscous flow over sphere, Re=100, Mach = 0.2
- 4th order RK time-stepping scheme
- Considered 3 grid types, each made up of one of the 3 element types
- Every effort was made to maximize performance of CPU code:
  - Intel Math Kernel Library (MKL) version 10.3 for dense MM
  - Optimized Sparse Kernel Interface (OSKI) for sparse MM
  - Cuthill-McKee renumbering of cells to maximize cache-hits
- All simulations use double precision math

GPU Implementation



Performance in Gflops of single GPU algorithm running on Tesla C2050

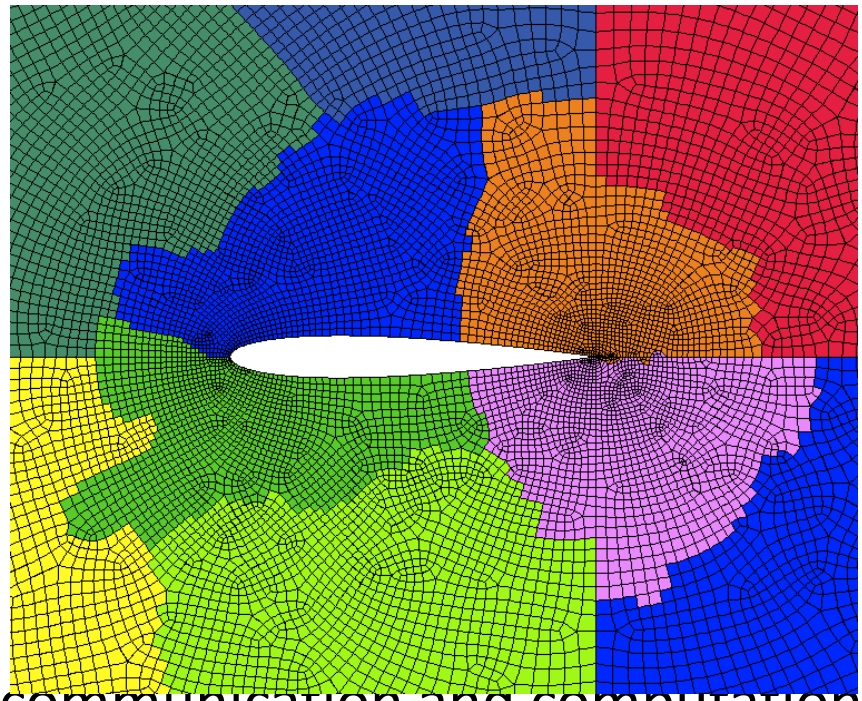
GPU Implementation



Speedup of the single-GPU algorithm (C2050) relative to a parallel computation on a quad-core Intel i7 930 @ 2.80GHz

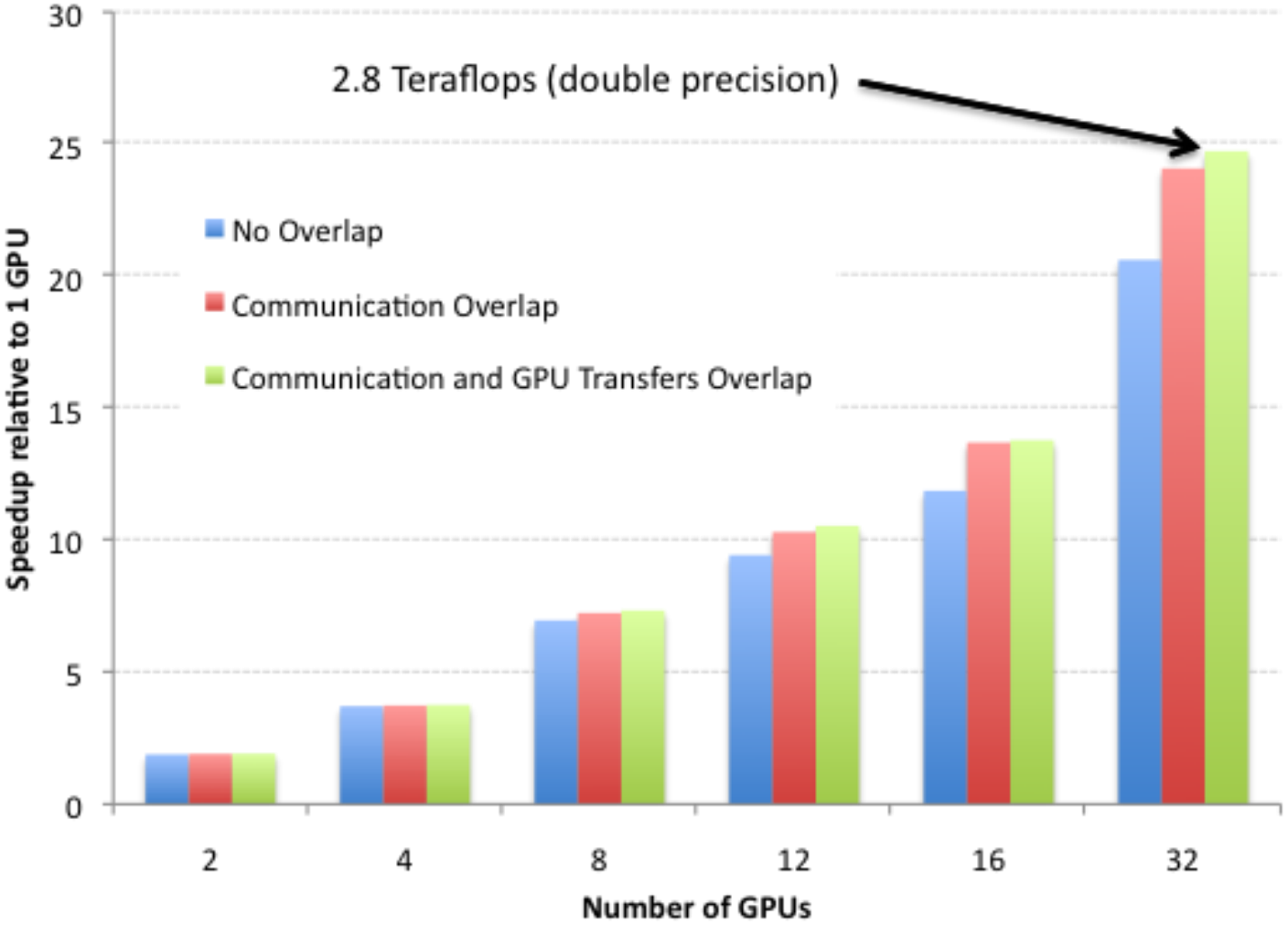
Multi-GPU Implementation

- Use mixed MPI-CUDA implementation to make use of multiple GPUs working in parallel
- Computational domain divided between GPUs using graph partitioning software ParMETIS



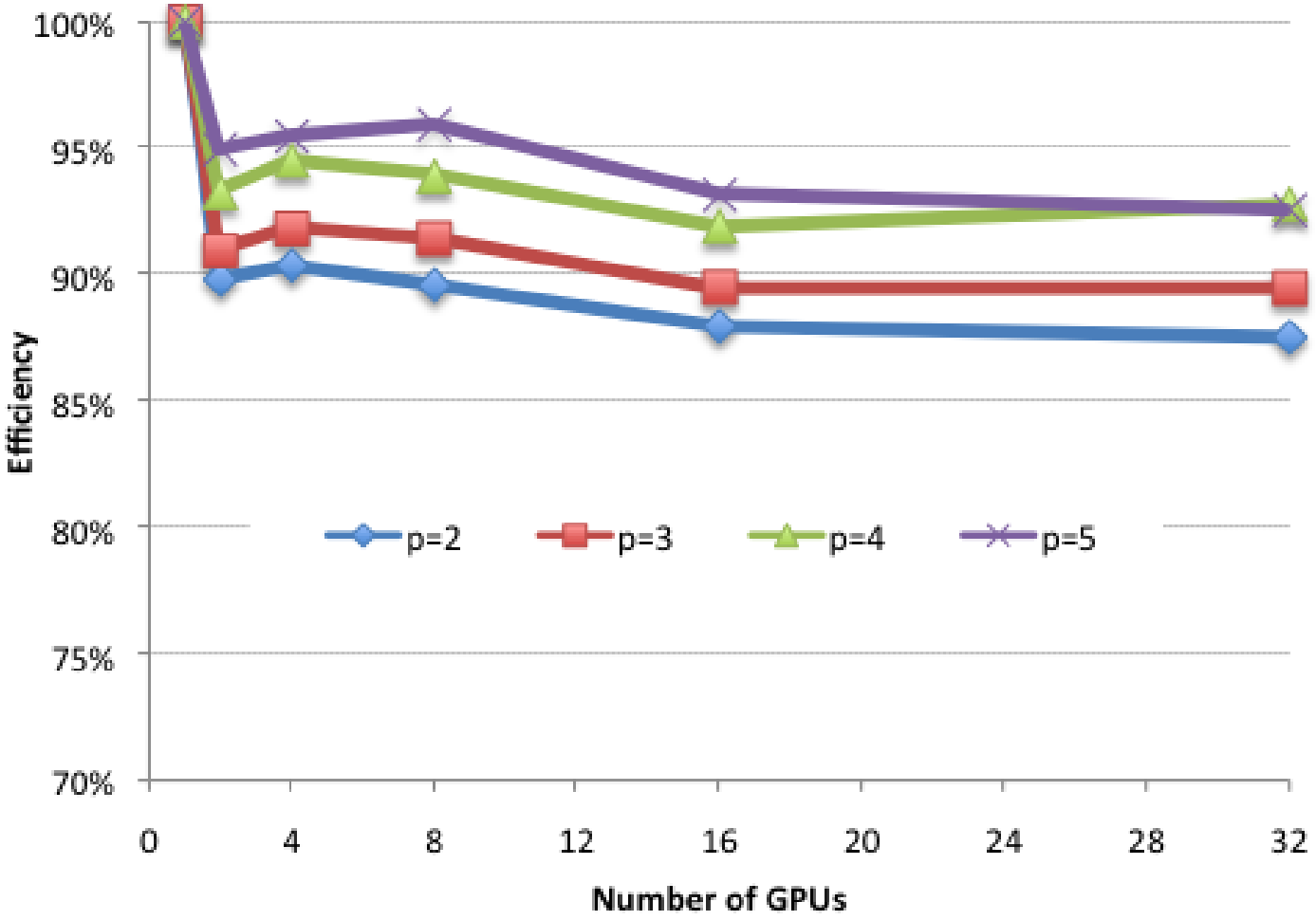
- Overlapping communication and computation using CUDA streams to achieve good performance

Multi-GPU Implementation



Speedup relative to 1 GPU versus the number of GPUs for a 6th order accurate simulation running on a mesh with 55947 tetrahedral elements

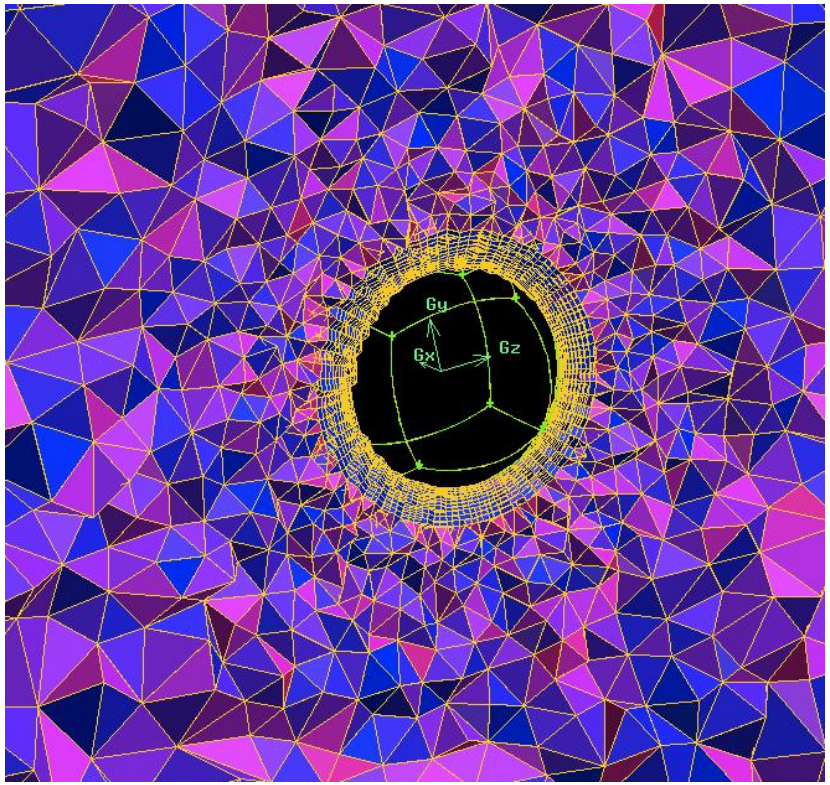
Multi-GPU Implementation



Weak Scalability of multi-GPU code: 27915 ± 1% Tets per GPU

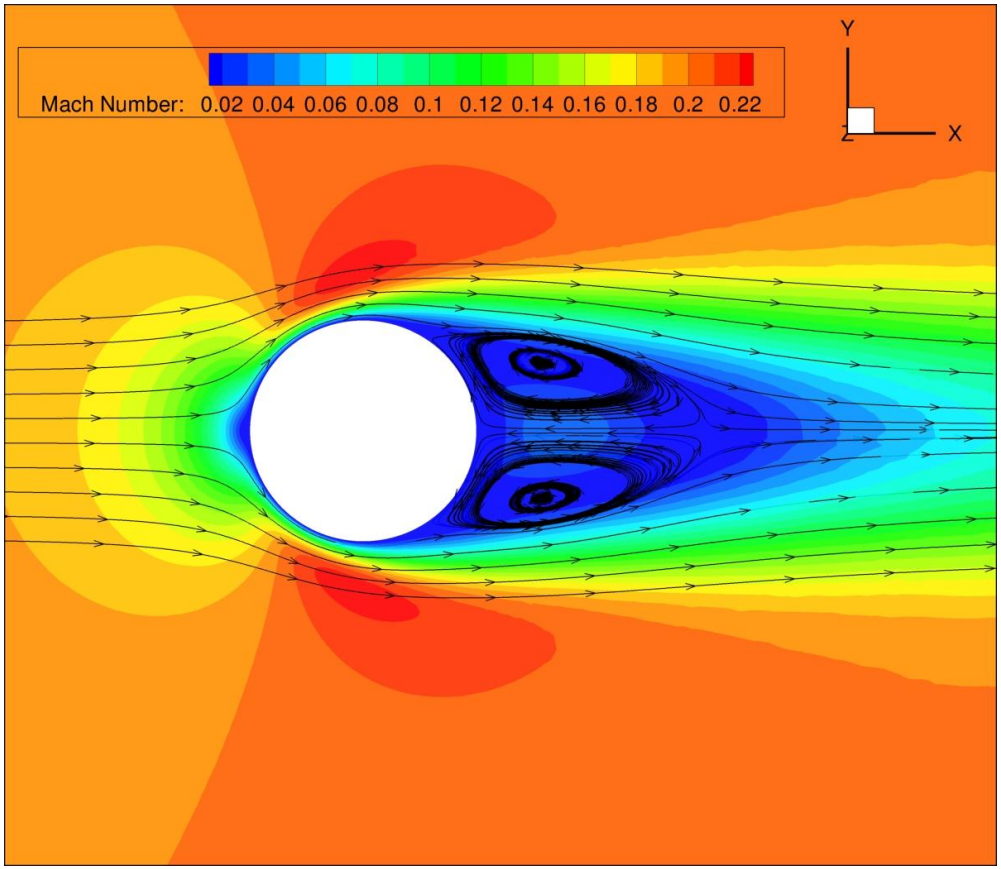
Applications

- Viscous flow over sphere at Reynolds 118, Mach=0.2
- 38,500 prisms and 99,951 tets , 4th order accuracy, 3.54 million DOFs
- Ran on desktop machine we built, 3 C2050 GPUs



Applications

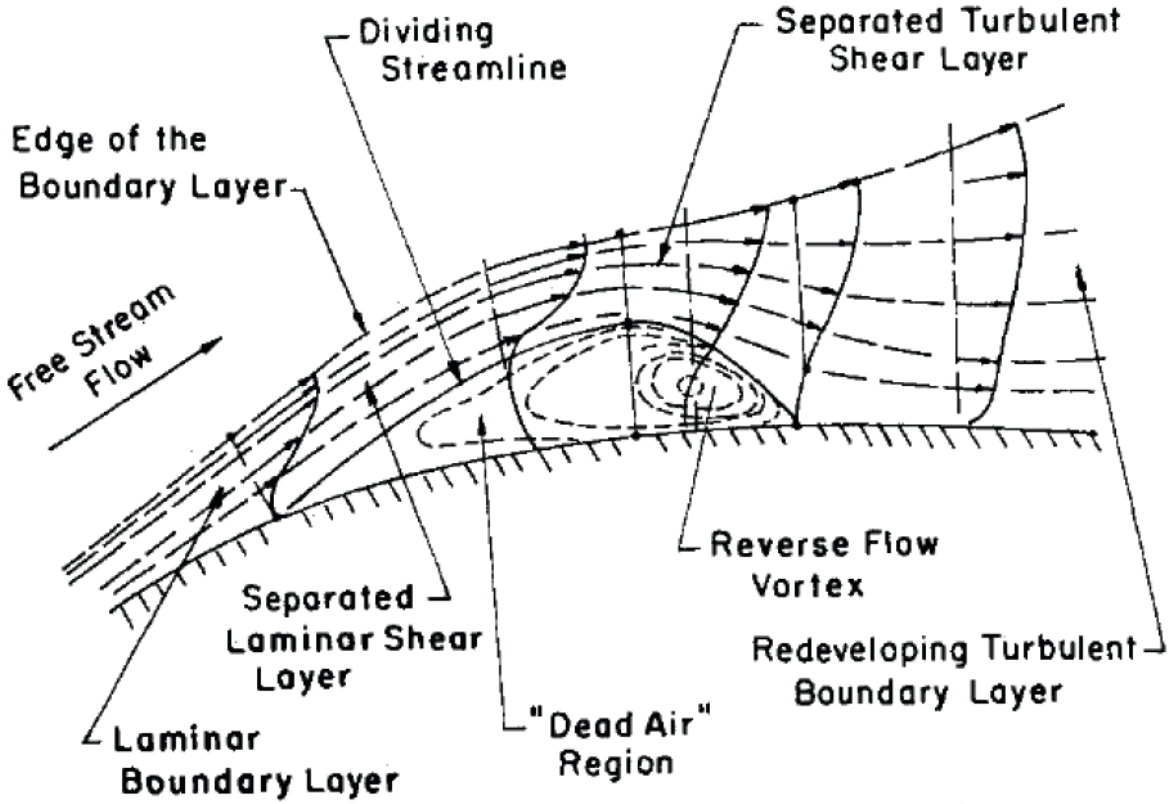
- 3 GPUs: same performance as 30 Xeon x5670 CPUs (180 cores)
- 3 GPUs personal computer: ~$10,000, easy to manage



Contours of Mach number for flow over sphere at Re=118, M=0.2

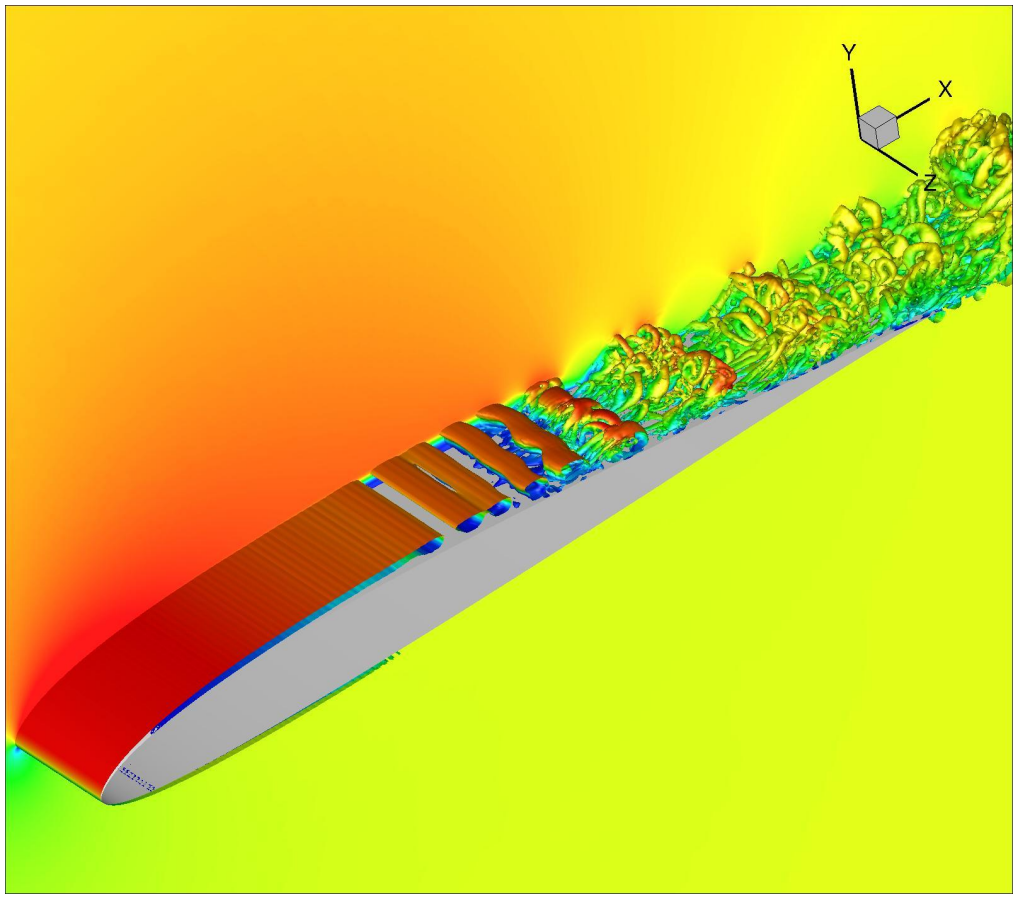
Applications

- At Reynolds number in range 10^4 to 10^5, flow over wings often characterized by formation of a Laminar Separation Bubble
- Important: birds and small UAVs fly in that regime
- Complex flow physics:

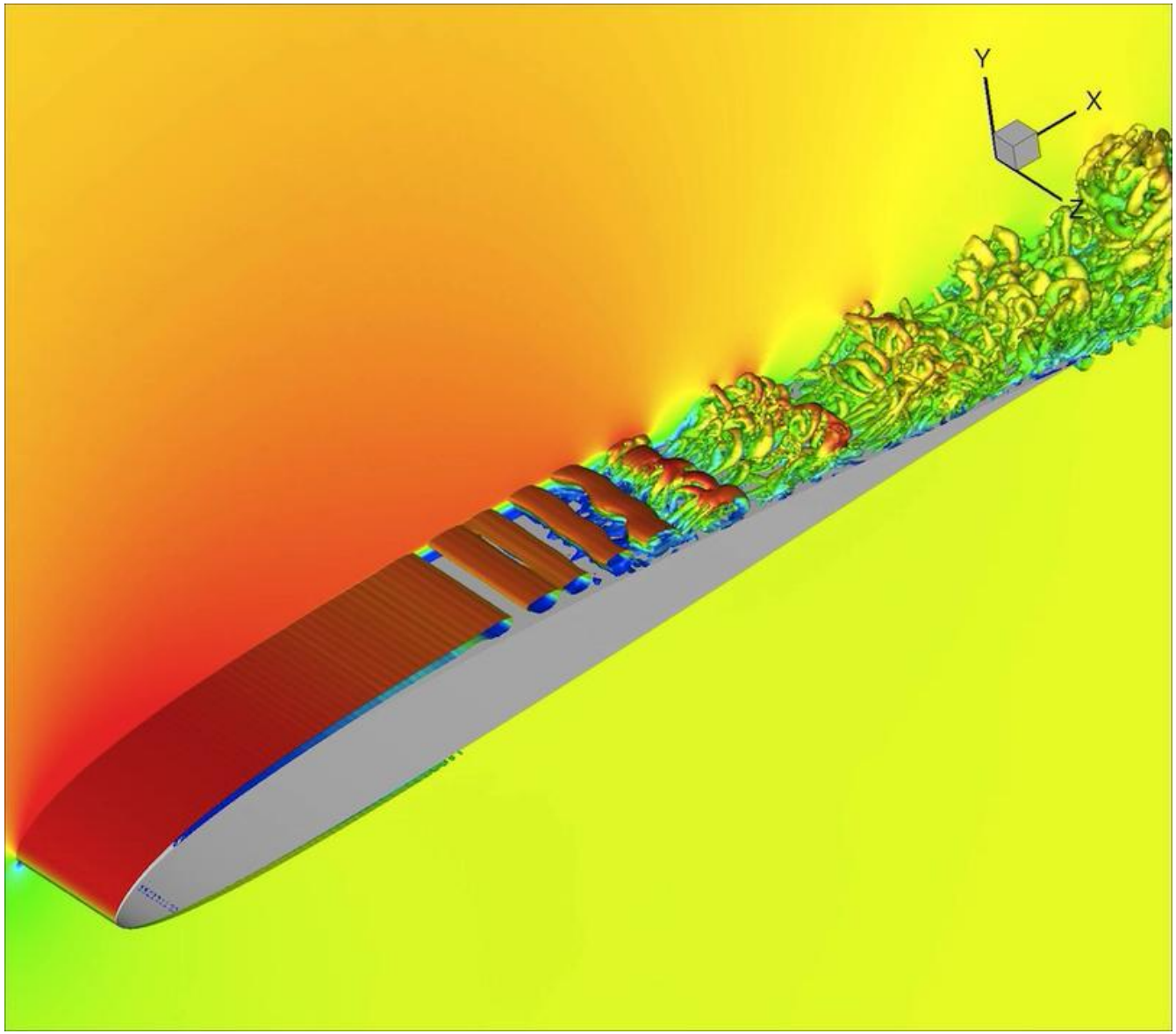


Applications

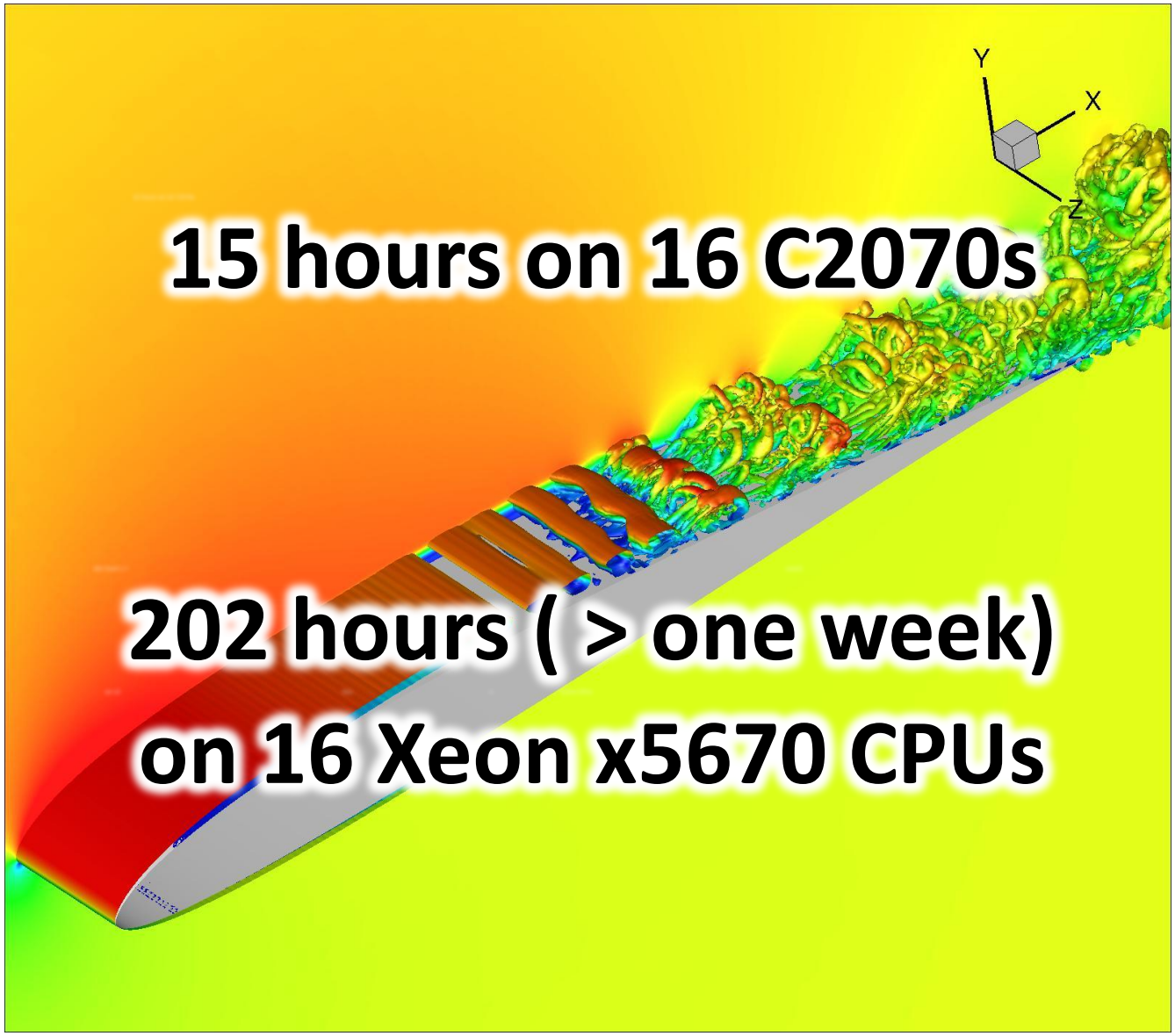
- Transitional flow over SD7003 airfoil, Re=60000, Mach=0.2, AOA=4°
- 4^{th} order accurate solution, 400000 RK iterations, 21.2 million DOFs



Applications



Applications



Conclusions

- Developed fast high-order CFD solver that can run on mixed unstructured grids on multiple GPUs
- GPUs enable simulation of previously intractable problems
- More than 100 Gigaflops on a workstation, few Teraflops on small GPU cluster
- Scaling demonstrated on up to 32 GPUs
- Next steps: LES models, more complex geometries

Acknowledgments

- Acknowledgments:
  - Peter Vincent, David Williams, Kui Ou, Yves Allaneau
  - NSF (Grants 0708071 and 0915006)
  - AFOSR (Grants FA9550-07-1-0195 and FA9550-10-1-0418)
  - Stanford Graduate Fellowship (SGF) Program
  - National Sciences and Engineering Council (NSERC) of Canada
  - Fonds Quebecois de la Recherche sur la Nature et les Technologies (FQRNT)
  - NSF Graduate Research Fellowship Program
  - NVIDIA

- Questions?

Patrice Castonguay pcasto2@stanford.edu