Application of GPU technology to OpenFOAM simulations

Jakub Poła, Andrzej Kosior, Łukasz Miroslaw

jakub.pola@vratis.com,
www.vratis.com
Wroclaw, Poland

Agenda

- Motivation
- Partial acceleration
  - SpeedIT
  - OpenFOAM SpeedIT Plugin
- Full acceleration
  - SpeedIT FLOW
  - Examples
- Summary

Problem

The more accurate models the more resources they require.

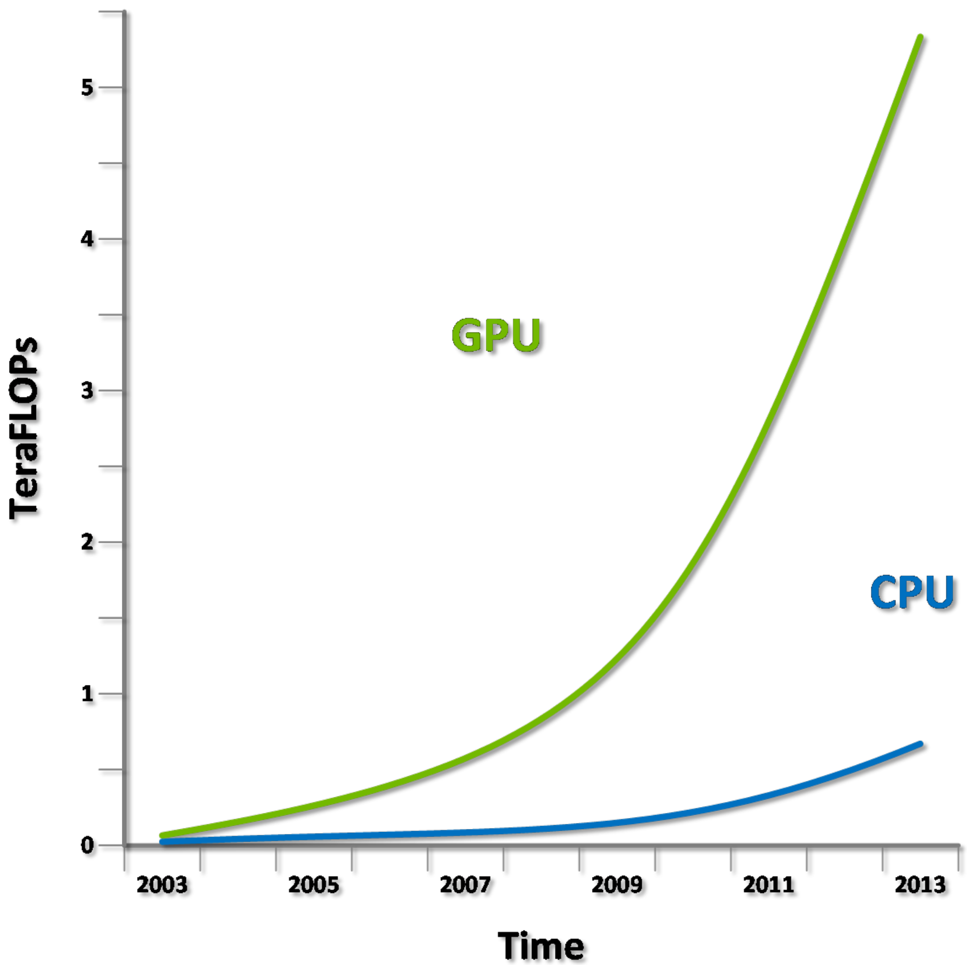
Solution #1

Use HPC CPU based systems

Solution #2

Unleash the computational power of the GPGPU

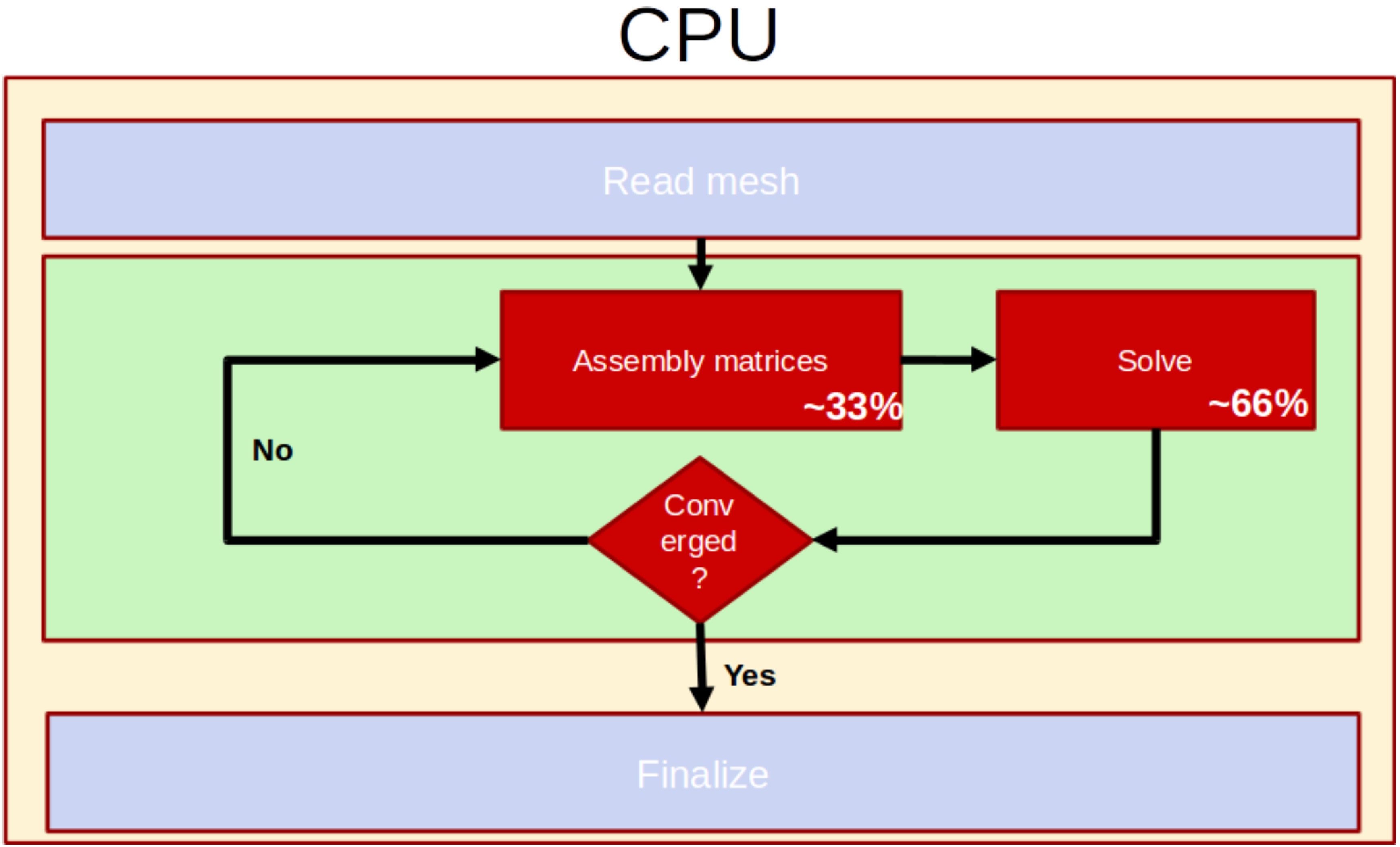
Why GPU?



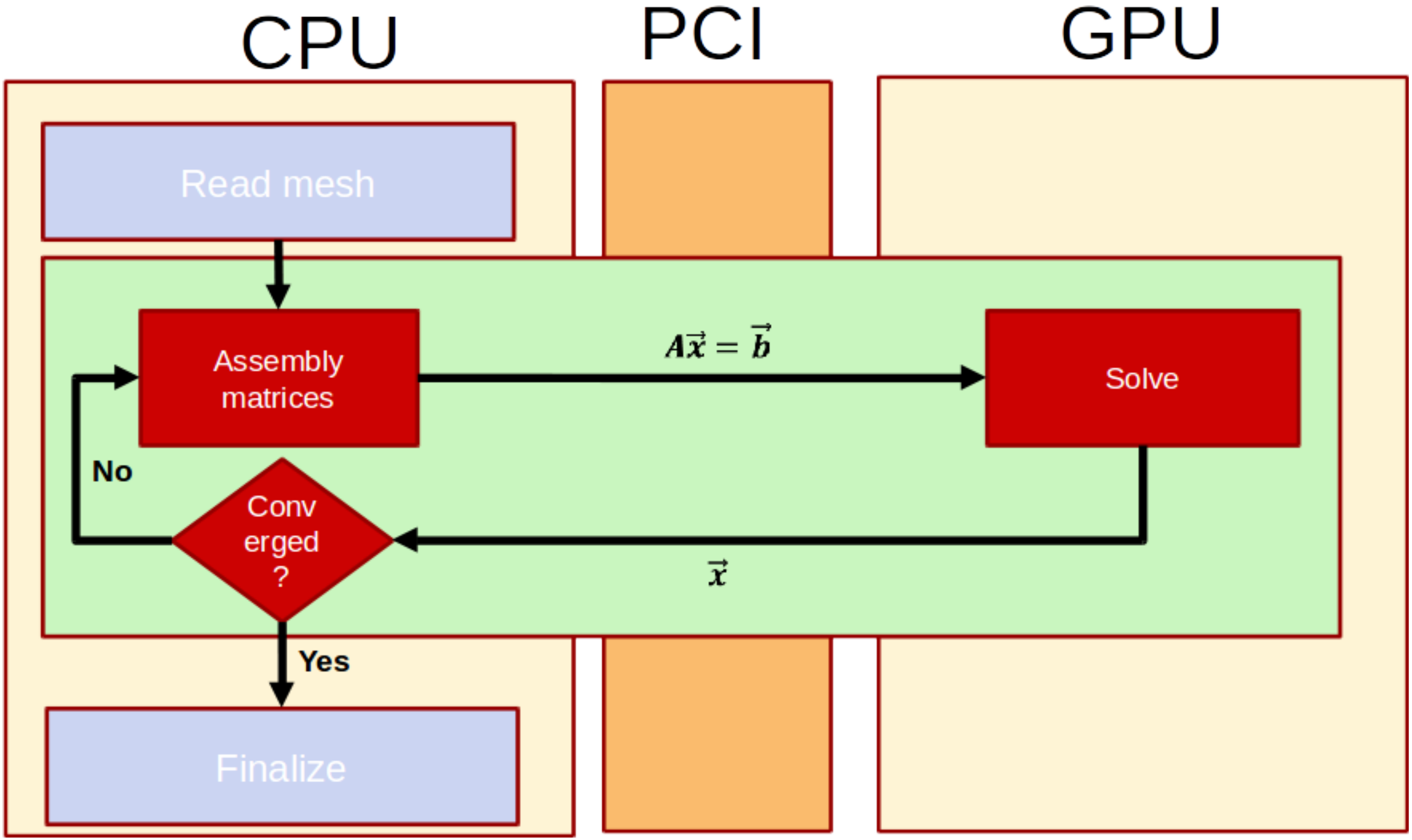
Why GPU?


GOOGLE BRAIN
1,000 CPU Servers
2,000 CPUs • 16,000 cores
600 kWatts
$5,000,000
300X energy efficiency
400X lower cost
Fits under a desk
GEFORCE GTX
GEFORCE GTX
GEFORCE GTX
1 Titan Z-Accelerated Server
3 Titan Zs • 17,280 cores
2 kWatts
$12,000

Case #1. Partial acceleration



Case #1. Partial acceleration



SpeedIT: Linear Algebra on GPU

- Solvers:
  - Conjugate Gradient.
  - Bi-Conjugate Gradient Stab.
- Preconditioners:
  - Diagonal.
  - Approximate Inverse.
  - Algebraic Multigrid with Smoothed Aggregation (CUSP).
- Support for Multi-GPU.
- Platforms:
  - OpenCL.
  - CUDA.

SpeedIT Integration with OpenFOAM

OpenFOAM plugin:

- libspeedit_plugin.so
- Conversion:
  - from LDU to CSR
- Solvers:
  - BiCGStab: SI_PBiCG
  - CG: SI_PCG
- Provides interfaces for multi-gpu calculations

contrlolDict:

```
libs(
"libspeedit_plugin.so"
"libspeedit.so"
)
```

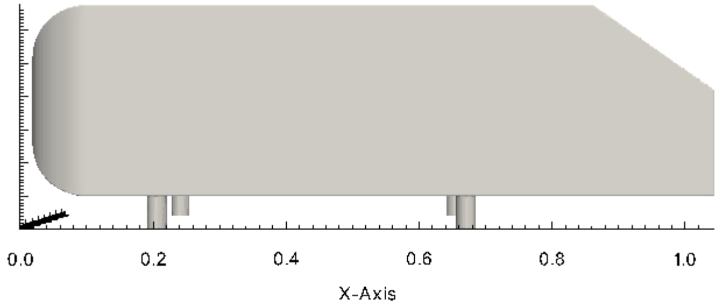
fvSolution:

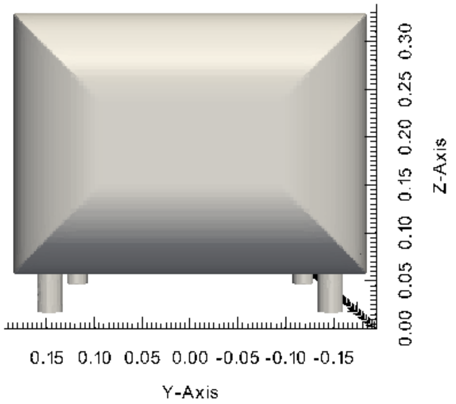
```
p  {
solver SI_PCG;
preconditioner
SI_AMG;
matrix CSR;
}
```

```
U {
solver SI_PBiCG;
preconditioner
SI_DIAGONAL;
matrix CSR;
}
```

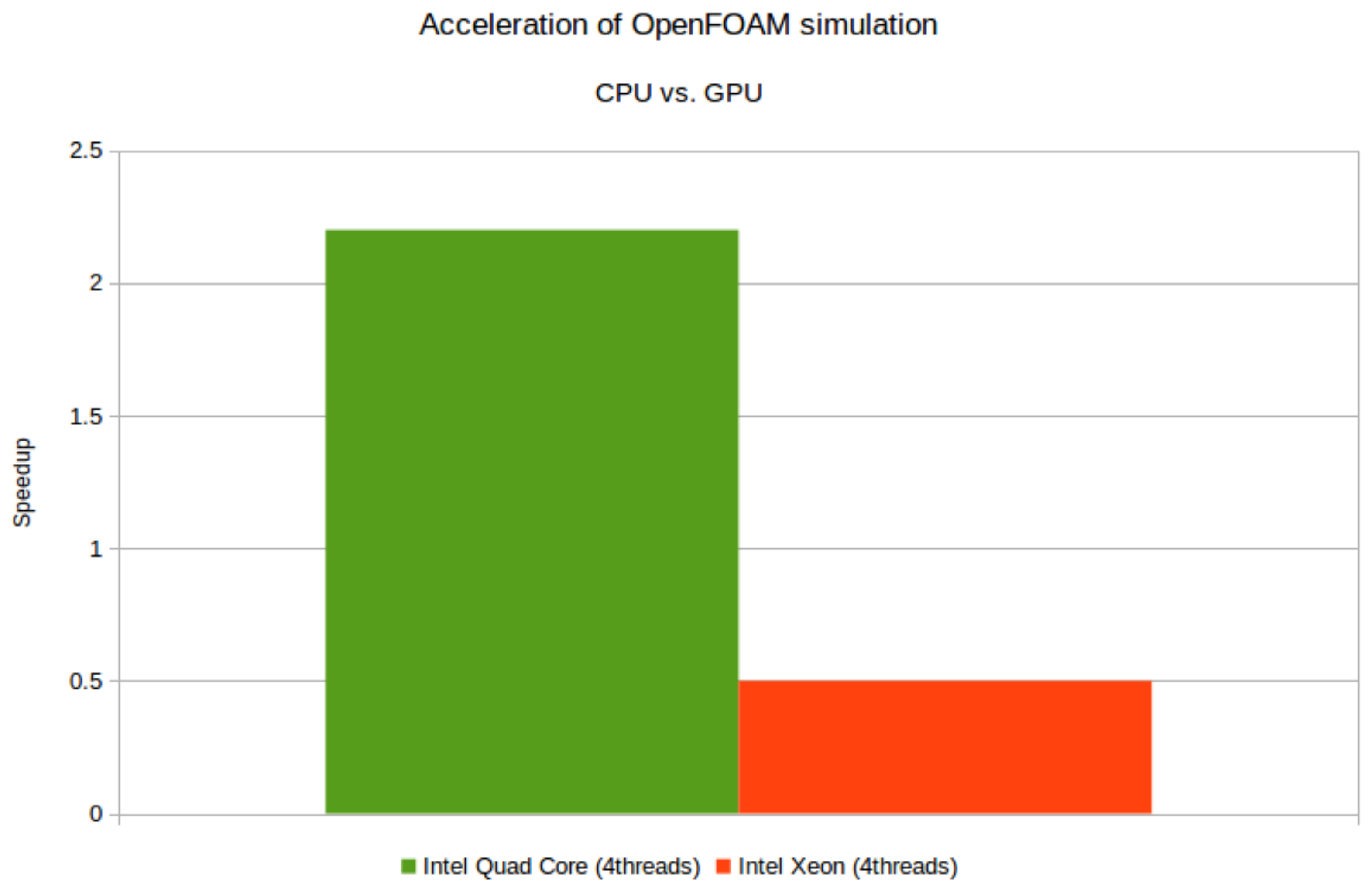
Case #1. Ahmed body

- Simulation parameters:
  - 1.37M cells
  - Stationary flow
  - Pressure solver / precond:
    - GAMG(CPU), PCG-AMG (GPU)
  - Velocity solver precond:
    - PBiCG - DILU(CPU), PBiCGStab Diagonal(GPU)
  -
- Hardware:
  - System #1: Intel Core 2 Quad, Tesla C2090 GPU
  - System #2: Intel Xeon, Tesla C2090 GPU

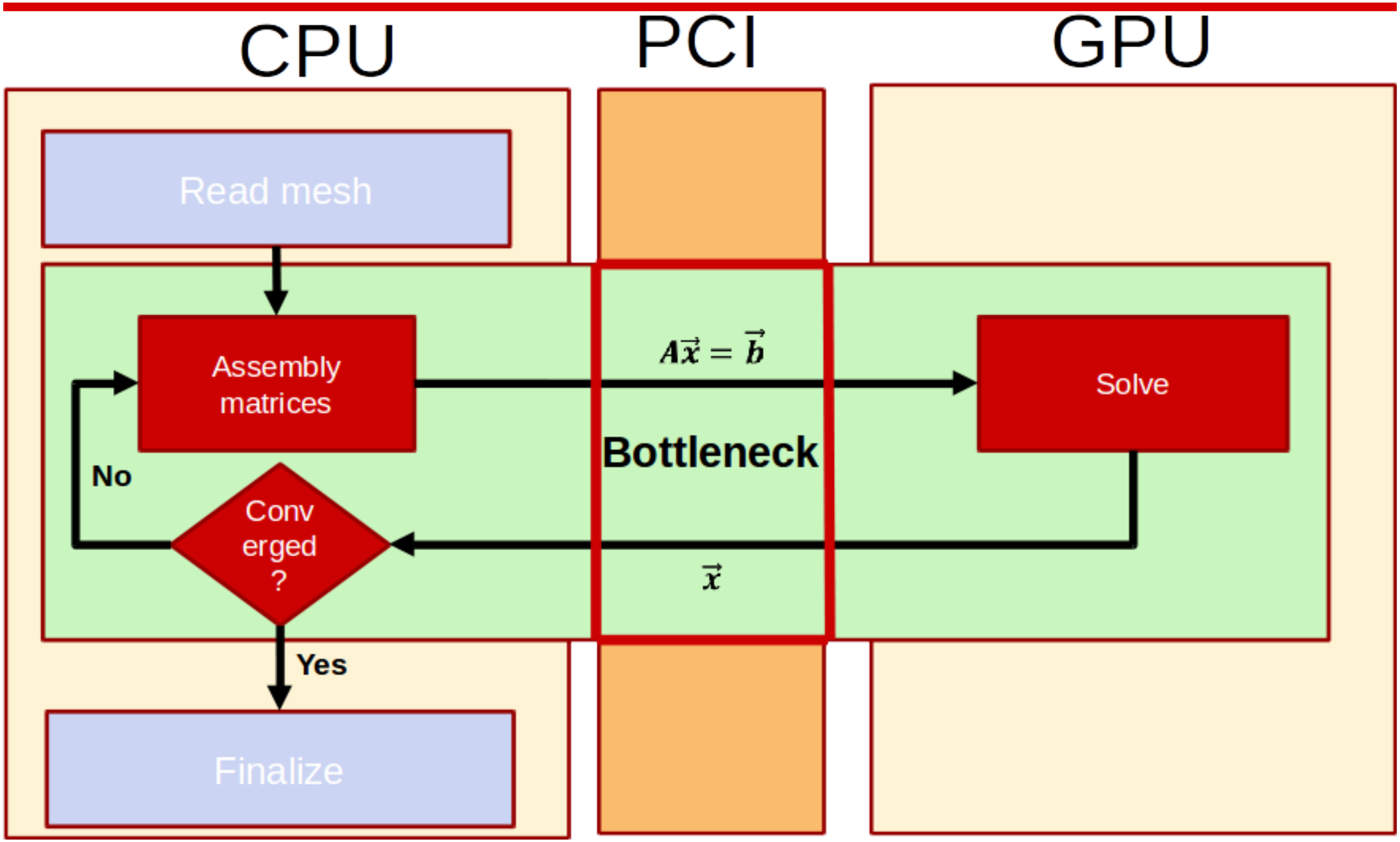




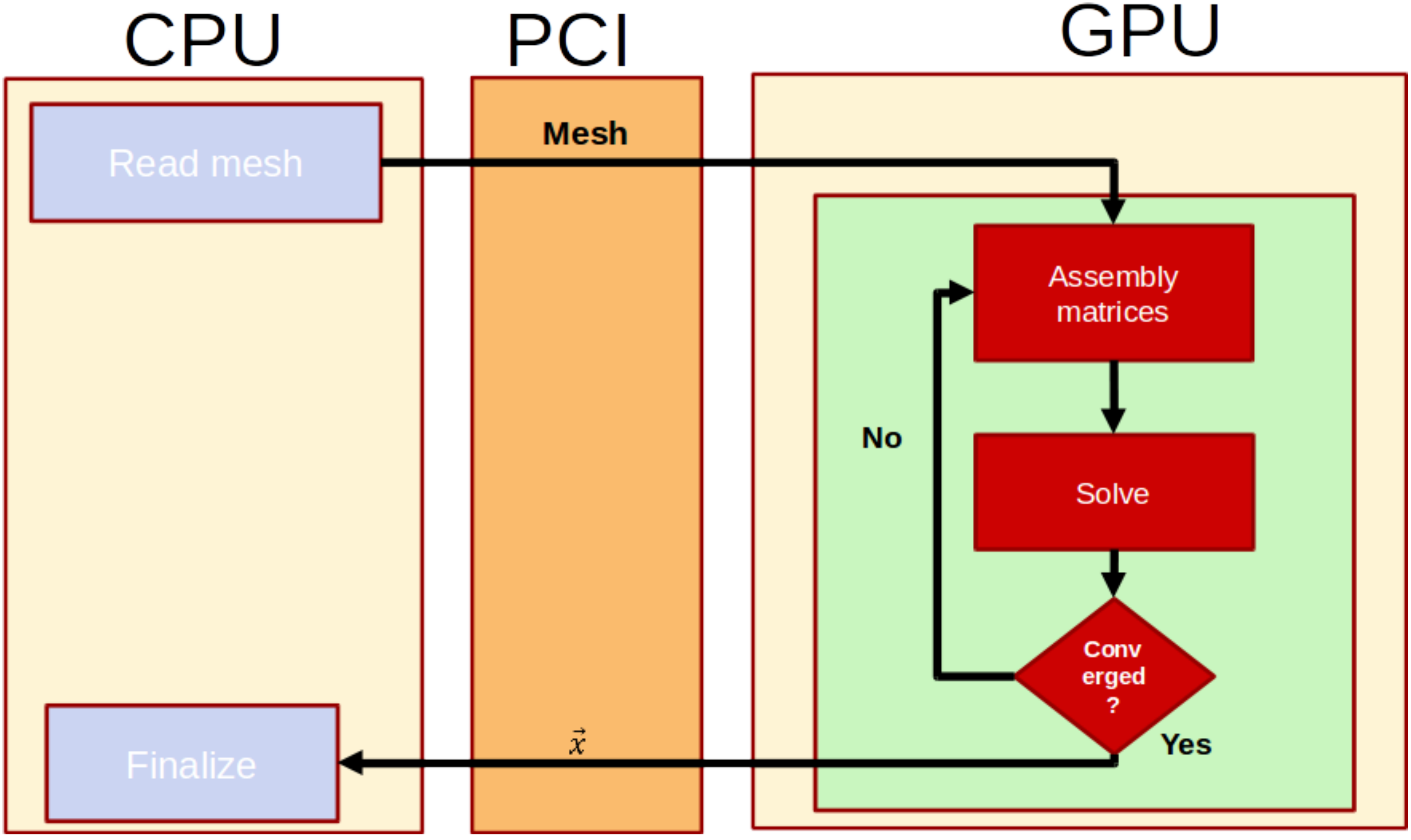
Partial acceleration: Results



Bottleneck: Memory transfer



Solution



CPU
PCI
GPU
Read mesh
Mesh
Assembly matrices
Solve
No
Conv erged ?
Yes
$\vec{x}$
Finalize

SpeedIT Flow

- Full GPU implementation of:
  - PISO
  - SIMPLE
  - Turbulence k-ω SST
- Boundary Conditions:
  - Zero gradient.
  - Time dependent fixed-value and slip
  - InletOutlet.
  - kqRWallFunction, omegaWallFunction, nutkWallFunction.
- Adjustable time step.
- Use OpenFOAM case definition for I/O
- Road Map:
  - Support for Multi-GPU.

SpeedIT Flow in action

Three simple steps:

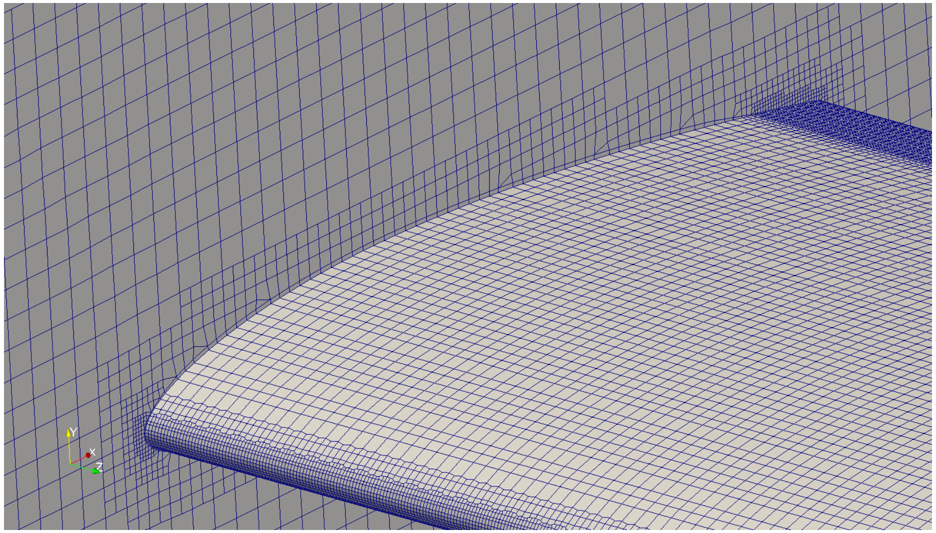
1. Generate the mesh using blockMesh and snappyHexMesh
2. Execute the converter to prepare the data in a GPU-friendly format.
3. Run the GPU based solver (gSIMPLE, gPISO)

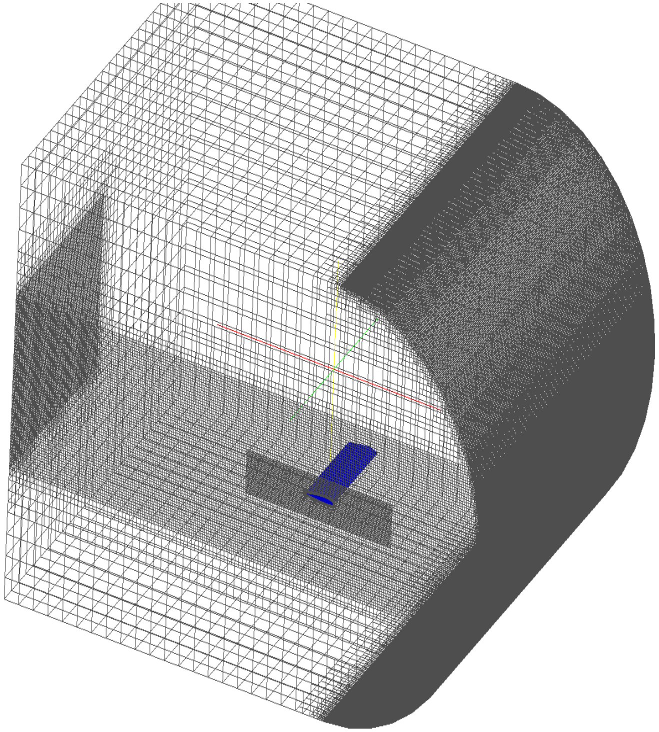
SpeedIT Flow in action: Configuration

- Steady State simulation, SIMPLE algorithm with *k*-ω SST turbulence model
- Meshing on CPU, Solving on GPU. Double Precision.
- External, turbulent flow at various speed.
- CPU Configuration:
  - Intel Xeon 5649 @ 2.53 GHz, 96 GB RAM
  - OpenFOAM 2.0.1 in parallel mode using 12 threads
    - Pressure solver: GAMG
    - Velocity, k, ω solvers: PBiCG with diagonal preconditioner
- GPU Configuration:
  - Intel Xeon 5649 @ 2.53 GHz, 96 GB RAM,
  - GPU: NVIDIA Quadro K6000, AMD W9100
  - SpeedIT FLOWCL: GSIMPLE solver
    - Pressure solver: PCG with AMG preconditioner
    - Velocity, k, ω solver: PBiCGStab with diagonal preconditioner

SpeedIT FLOW

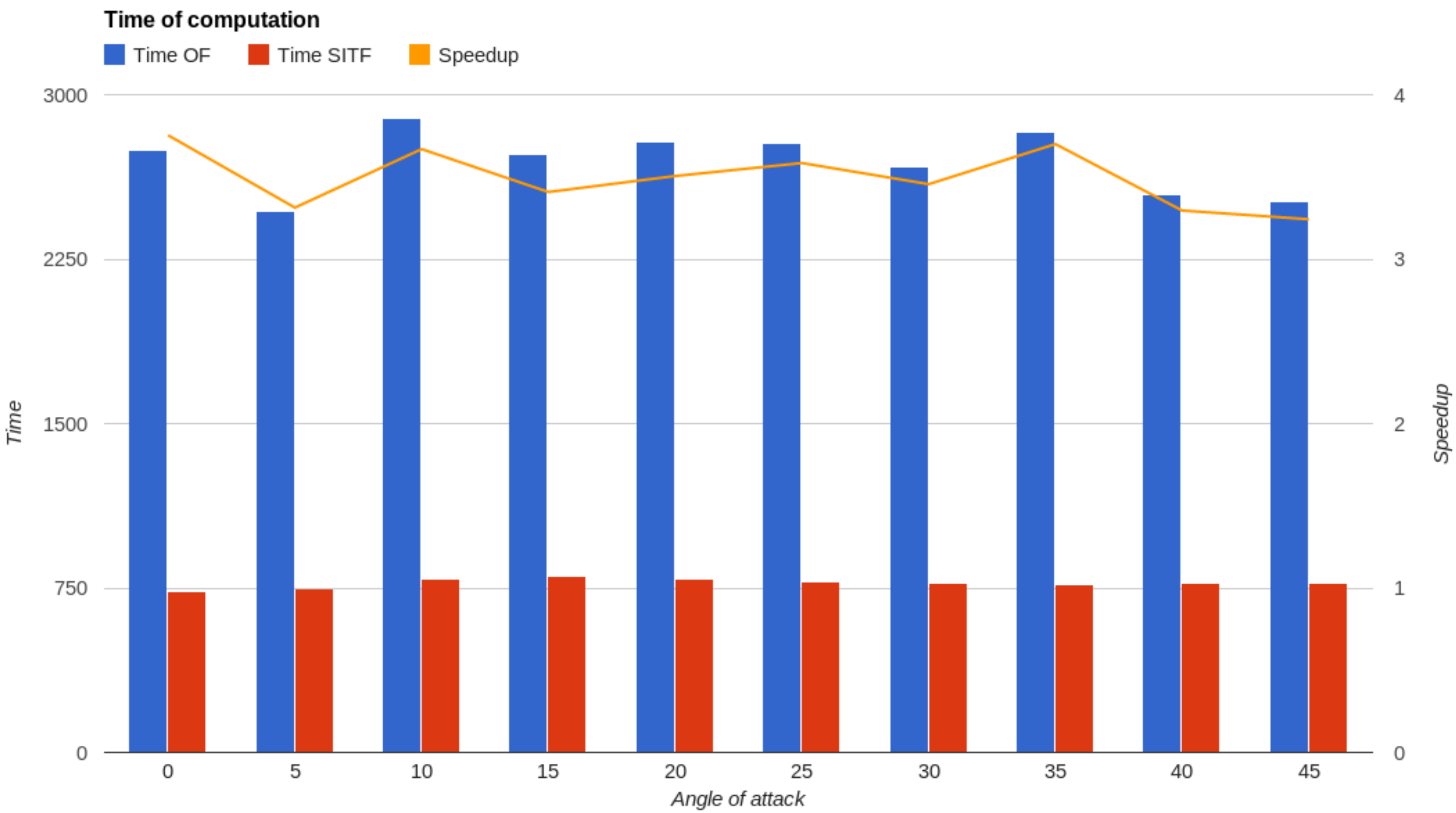
Case #2. NACA 2412 airfoil



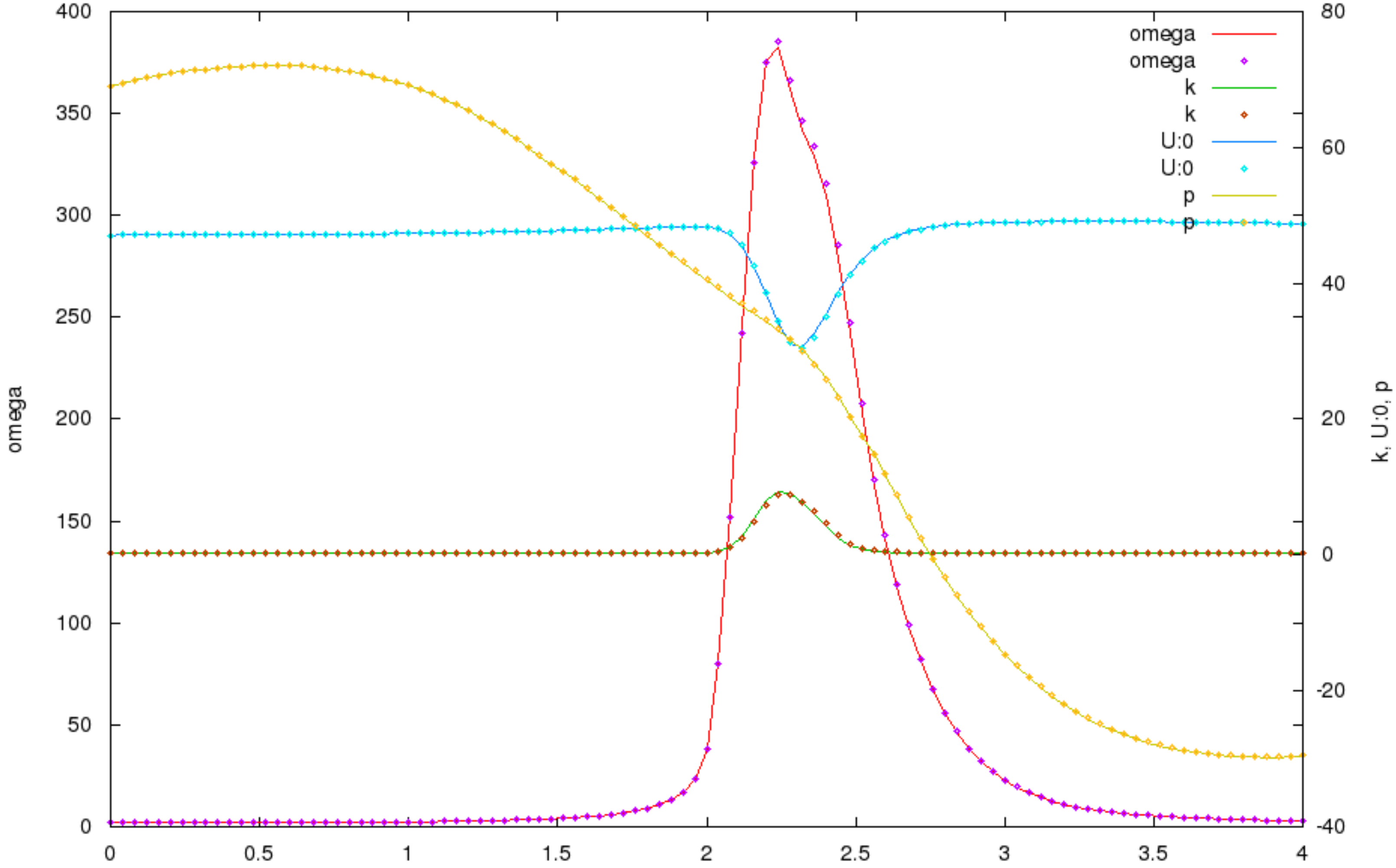


- chord - $c = 1m$, taper ratio - $c_{tip}/c_{root} = 1$, wing span - $b = 6m$ - span, wing area - $S = 6m^2$.
- 3401338 (3.4M) cells
- Different angles of attack from 0 to 45 degrees with a 5 degree step

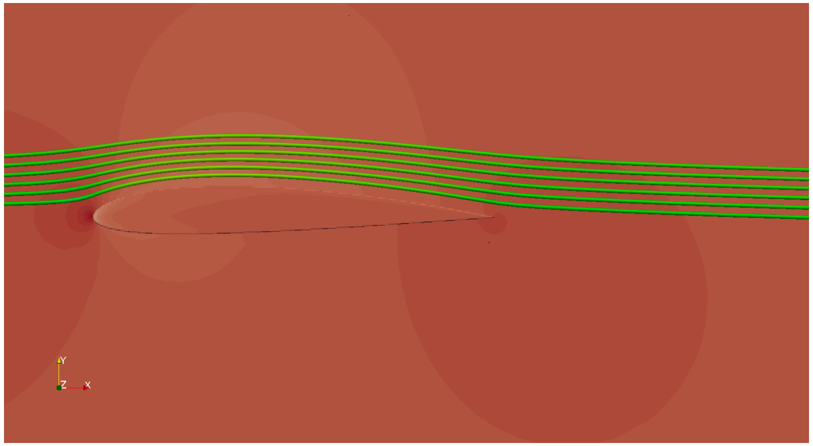
Case #2. Performance results

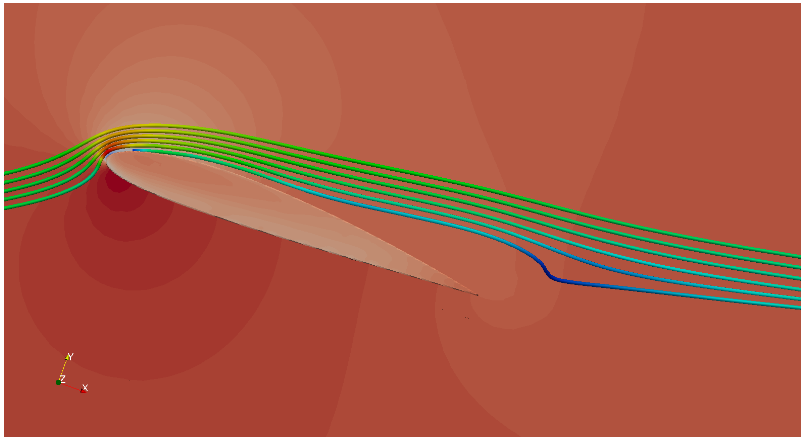


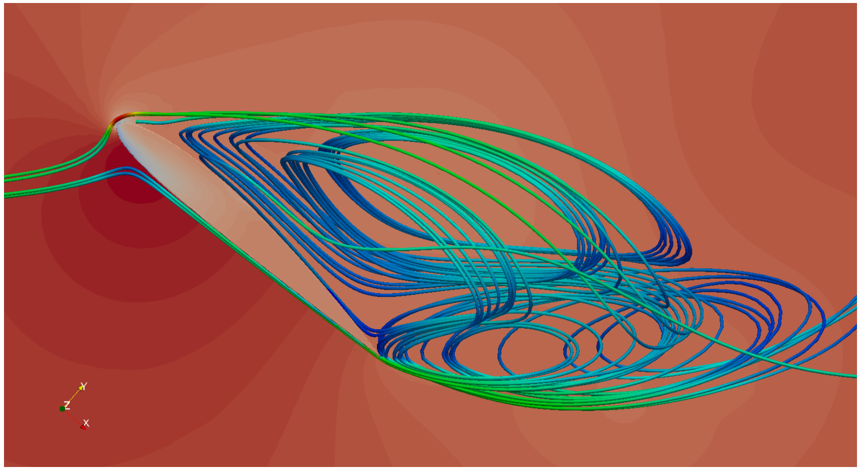
Case #2. Numerical results



Case #2. Visualisation







Cutting edge hardware

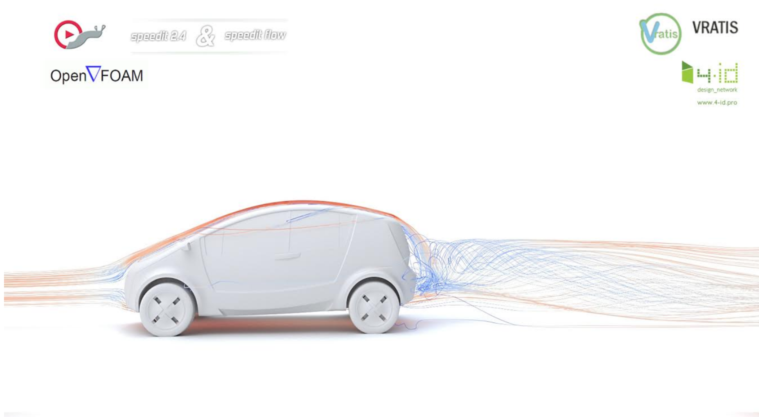
AMD FirePro S9150

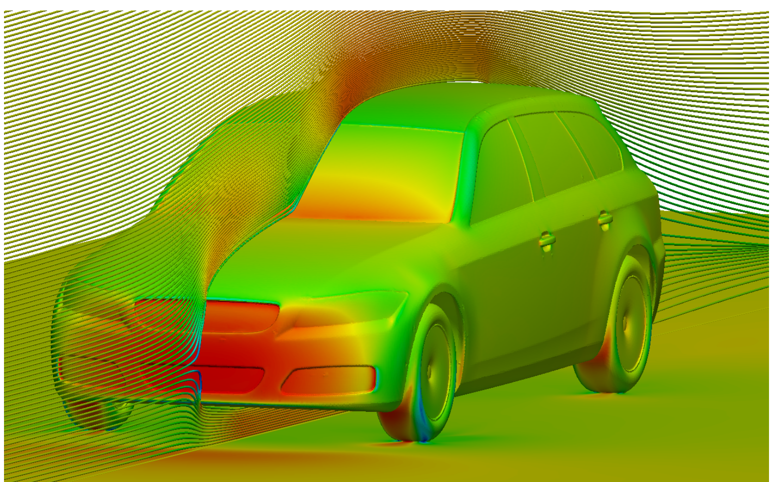
- 2.53 TFLOPS of peak double-precision performance
- 16GB GDDR5 memory
- 235W at maximum power

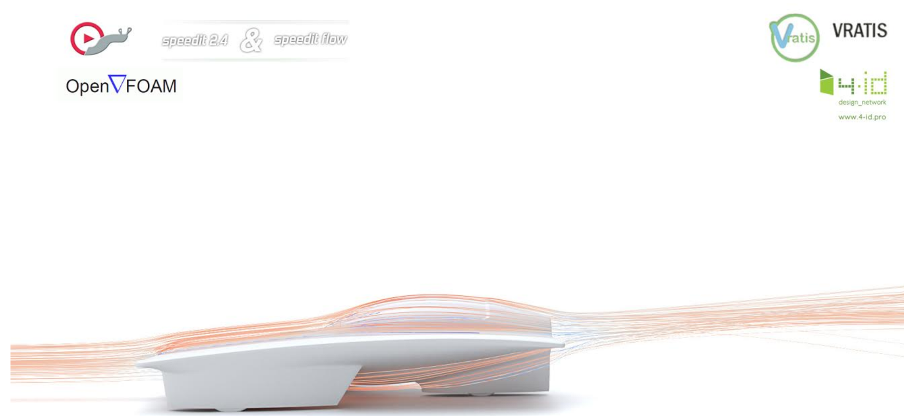
NVIDIA K6000

- 1.4 TFLOPS of peak double-precision performance
- 12GB GDDR5 memory
- 225W at maximum power

Case #3. Cars

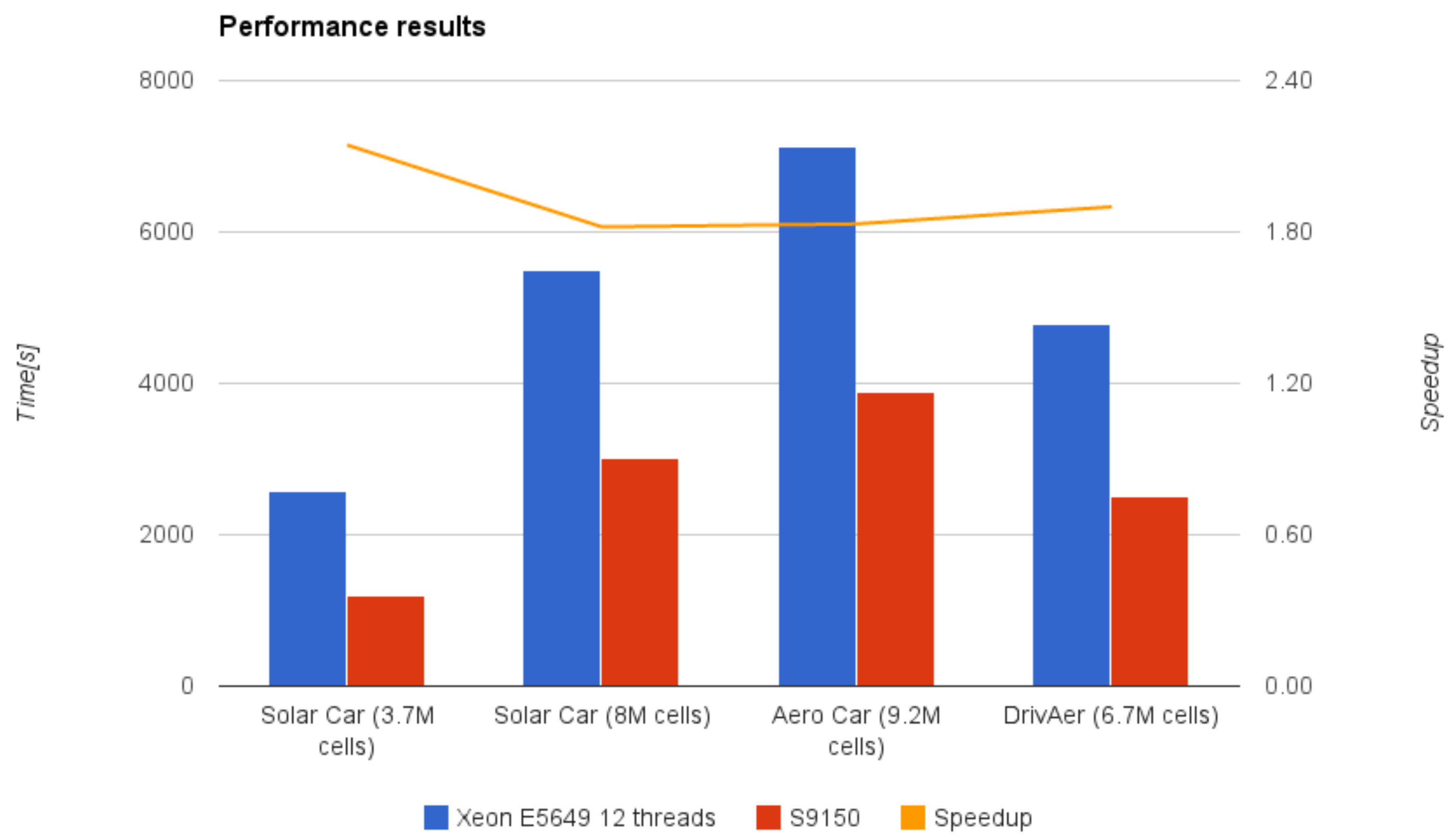




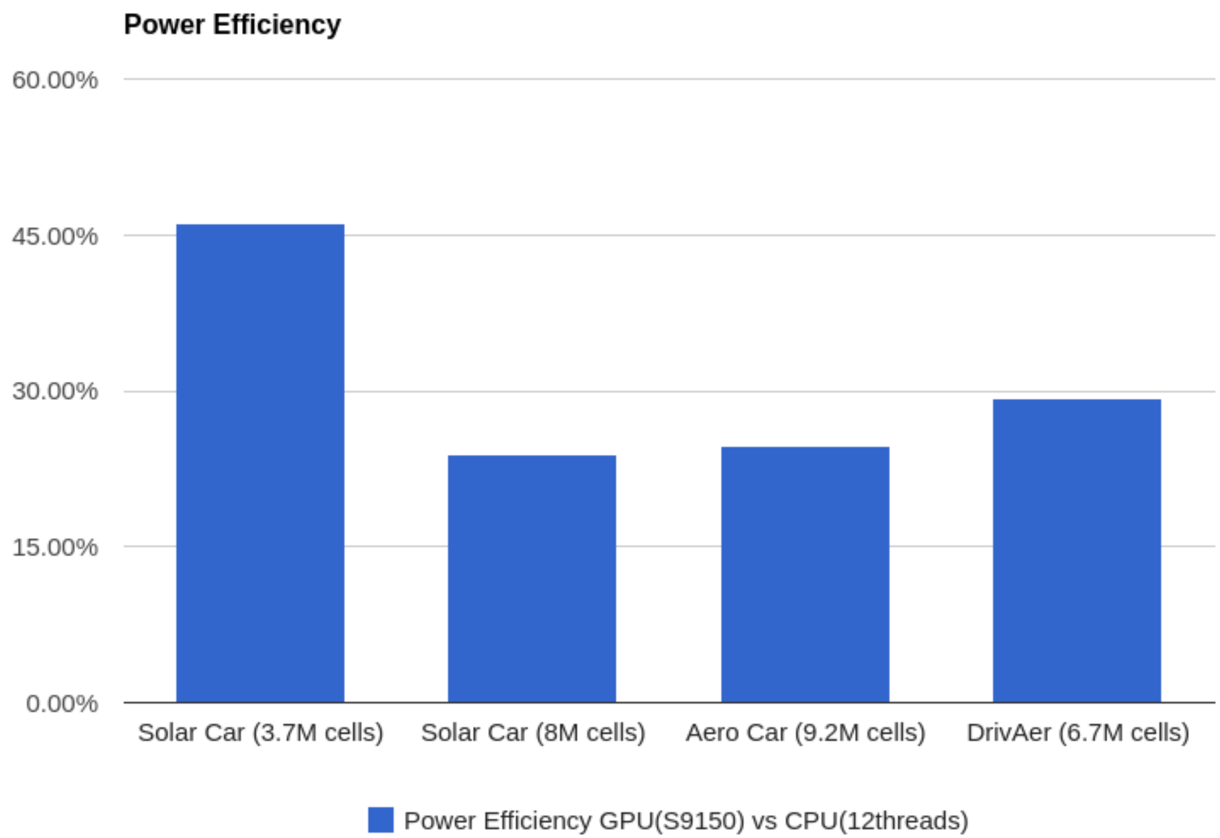


Origin of models: 4-ID Network, TUM (Technische Universität München)

Case #3. Performance results (AMD)



Case #3. Power consumption



Summary

1. Use of GPU to accelerate OpenFOAM simulation is profitable!
   - Speedup from x1.8 up to x3.5
   - Energy consumption is lower.
2. Use your GPU as external accelerator.
   - GPU uses 1 CPU thread.
   - Rest of the cores can be utilized for other tasks i.e for another GPU simulations.
3. With SpeedIT Flow w can:
   - Simulate flows on fully unstructured meshes up to 9M cells on single GPU
   - Accelerate solution of Navier - Stokes equations for steady-state and transient cases using SIMPLE and PISO algorithms
   - Solve the laminar and turbulent cases using k-w SST model
   - Use OpenFOAM as a standard for case definition

Questions?
Comments?

Jakub Pola
jakub.pola@vratis.com
www.vratis.com

Acknowlededemnents:

Vratis: Michal Cieslak, Andrzej Kosior, Lukasz Miroslaw

Wroclaw Uni.: Zbigniew Koza

Wroclaw Uni. of Technology: Tadeusz Tomczak