GPU Acceleration of Computational Fluid Dynamics (CFD) in Industrial Applications using Culises and aeroFluidX

GPU Technology Conference, March 25 2014, San Jose, CA
Content

- Introduction – potential of GPU-computing for CFD
- Short summary of **Culises**
  - Hybrid GPU-CPU approach for partially accelerated CFD applications
  - Industrial problem set and achievable speedups
- **aeroFluidX** – fully ported flow solver on GPU
  - Technical approach
  - Problem set and achievable speedups
- Conclusions and future roadmaps for Culises & aeroFluidX
GPU-computing for CFD
Potential

Automotive example: Car-truck interference

OpenFOAM®: simpleFoam solver
120M computing cells

Simulation time:
Medium CPU cluster:
22 dual-CPU socket blades:
→ 44 CPUs of type Sandy Bridge Xeon E5-2650, 8-core
→ runtime ≈ 1 weekend

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel E5-2650 V2</td>
<td>1000</td>
<td>166</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>Nvidia K40</td>
<td>3500 (3.5x)</td>
<td>1430 (8.6x)</td>
<td>288 (4.8x)</td>
<td>235 (2.35x)</td>
</tr>
</tbody>
</table>

Computing platform

<table>
<thead>
<tr>
<th>Hybrid CPU-GPU</th>
<th>Theoretical peak performance</th>
</tr>
</thead>
</table>
| 22 blades equipped with 2 x Intel Xeon E5-2650 V2
+ blade hardware: mainboard, memory, power supply ...
+ air-conditioned room required | 7304 Gflops (4400 Watt) |
| 44k € for CPUs only (Q1/2014) | 332 Gflops +5720 Gflops |
| 18k € (Q1/2014) | 7482 Gflops (1140 Watt) |
Library Culises
Concept and Features

Simulation tool e.g. OpenFOAM®

- **State-of-the-art solvers** for solution of linear systems
  - Multi-GPU and multi-node capable
  - Single precision or double precision available

- **Krylov subspace methods**
  - CG, BiCGStab, GMRES for symmetric/non-symmetric matrices
  - Preconditioning options
    - Jacobi (Diagonal)
    - Incomplete Cholesky (IC)
    - Incomplete LU (ILU)
    - Algebraic Multigrid (AMG), see below

- **Stand-alone multigrid method**
  - Algebraic aggregation and classical coarsening
  - Multitude of smoothers (Jacobi, Gauss-Seidel, ILU etc.)

- **Flexible interfaces** for arbitrary applications e.g.: established coupling with OpenFOAM®

**Culises** = **CUDA Library for Solving Linear Equation Systems**

See also [www.culises.com](http://www.culises.com)
GPU Acceleration of CFD in Industrial Applications using Culises and aeroFluidX

**Library Culises**

**Hybrid CPU-GPU scenario (MPI+Cuda)**

OpenFOAM®

**MPI-parallelized** CPU implementation based on domain decomposition

OpenFOAM:

- CPU 0
- CPU 1
- CPU 2

Linear system \( Ax=b \)

Processor partitioning

Solution \( x \)

Application interface

**Interface:**
- cudaMemcpy(....)
- cudaMemcpyHostToDevice
- cudaMemcpyDeviceToHost

Culises dynamic library:
Solves linear system(s) on multiple GPUs

Culises:
- PCG
- PBiCG
- AMGPCG

MPI-parallel assembly (discretization etc.) of system matrices remains on CPUs

**FluiDynata**

INNOVATIVE ENGINEERING AND BEYOND

GTC 2014

Slide 5
Culises

Hybrid parallel approach

1-1 link between MPI-process/rank and GPU
→ CPU-based partitioning equals GPU partitioning
→ peak performance of CPU << peak performance of GPU
→ under-utilization of GPUs
→ Bunching of MPI-ranks required

HyperQ (Nvidia Kepler: max. 32 ranks linked to one GPU)

Serial kernel execution

Concurrent kernel execution

Simulation time saved
Culises
Hybrid parallel approach

Two options:

a) HyperQ + multi-process service

b) Minimize communication: Stick together linear systems from multiple MPI-ranks and solve consolidated system

option a) multi-process service

\[
\begin{align*}
\text{matrix } A_0 : & \quad x_0 = b_0 \\
\text{matrix } A_1 : & \quad x_1 = b_1 \\
\text{matrix } A_2 : & \quad x_2 = b_2
\end{align*}
\]

option b) one kernel only (multi-level partitioning for best performance)

\[
\begin{align*}
\text{matrix } A : & \quad x = b
\end{align*}
\]
Only requirement: Use **same convergence criterion** for both, CPU and GPU linear solver

*Example: Airfoil*

- **Solver:** simpleFoam
- **Pressure solver accelerated with Culises**

**Validation with OpenFOAM®**

- **Residual pressure equation**
- **Lift coefficient**
  - OF stand alone (CPU only)
  - Culises (CPU+GPU)
- **Drag coefficient**
  - Simple iterations
Culises

Benchmarks

- CFD solver OpenFOAM® V2.2.2
- Fair comparison between linear solver of OpenFOAM® and Culises
  - Satisfy same tolerance for norm of residual
  - Choose best linear solver on CPU side vs. best linear solver on GPU
    Krylov method or Multigrid method or combination of both?

- Hardware:
  - CPU: Intel Sandy Bridge E5-2650, @ 2.0 GHz (8-core)
  - GPU: Nvidia Tesla K40 (12GB)

Theory: Linear solver scaling (Stüben 2002)

Common knowledge: some multilevel/multigrid approach is definitely needed, either
- Multigrid as stand-alone solver, or
- Multigrid as preconditioner
Automotive: generic car shape model

- simpleFoam solver from OpenFoam®
  - Steady-state (SIMPLE\textsuperscript{1}) method
  - Linear solver settings
    - Only linear system for pressure correction accelerated by Culises
    - Linear systems for velocities solved CPU-only cause CPU-GPU overhead outbalances GPU-acceleration

- \(k-\omega\) SST turbulence model (CPU-only)
## Culises

### Multi-GPU runs

- **Weak scaling analysis:** grid series 12.5M, 25M, 37.5M, 50M cells
- **CPU linear solver for pressure:** geometric-algebraic multigrid (GAMG) of OpenFoam®
- **GPU linear solver for pressure:** AMG preconditioned CG (AMGPCG) of Culises
- 100 SIMPLE iterations

<table>
<thead>
<tr>
<th>Grid cells</th>
<th>CPU cores</th>
<th>GPU cores</th>
<th>Linear solve time [s]</th>
<th>Total simulation time [s]</th>
<th>Speedup linear solver</th>
<th>Speedup total simulation</th>
<th>Fraction f = linear solve time/total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.5M</td>
<td>8</td>
<td>0</td>
<td>873</td>
<td>2373</td>
<td>---</td>
<td>---</td>
<td>0.37</td>
</tr>
<tr>
<td>12.5M</td>
<td>8</td>
<td>1</td>
<td>380</td>
<td>1867</td>
<td>2.30</td>
<td>1.27</td>
<td>---</td>
</tr>
<tr>
<td>25M</td>
<td>16</td>
<td>0</td>
<td>886</td>
<td>2363</td>
<td>---</td>
<td>---</td>
<td>0.37</td>
</tr>
<tr>
<td>25M</td>
<td>16</td>
<td>2</td>
<td>452</td>
<td>1934</td>
<td>1.96</td>
<td>1.22</td>
<td>---</td>
</tr>
<tr>
<td>37.5M</td>
<td>24</td>
<td>0</td>
<td>967</td>
<td>2504</td>
<td>---</td>
<td>---</td>
<td>0.39</td>
</tr>
<tr>
<td>37.5M</td>
<td>24</td>
<td>3</td>
<td>518</td>
<td>2001</td>
<td>1.87</td>
<td>1.25</td>
<td>---</td>
</tr>
<tr>
<td>50M</td>
<td>32</td>
<td>0</td>
<td>1062</td>
<td>2614</td>
<td>---</td>
<td>---</td>
<td>0.41</td>
</tr>
<tr>
<td>50M</td>
<td>32</td>
<td>4</td>
<td>557</td>
<td>2067</td>
<td>1.91</td>
<td>1.26</td>
<td>---</td>
</tr>
</tbody>
</table>
Culises

Multi-GPU runs

- Weak scaling analysis: grid series 12.5M, 25M, 37.5M, 50M cells
- CPU linear solver for pressure: geometric-algebraic multigrid (GAMG) of OpenFoam®
  GPU linear solver for pressure: AMG preconditioned CG (AMGPCG) of Culises
- 100 SIMPLE iterations

Weak scaling efficiency (=100%*t(1 GPU)/t(n GPUs))

- CPU
- GPU
Culises

Multi-GPU runs

- Automotive industrial setup (Japanese OEM)
- Same solver applied as with DrivAER case but strong scaling analysis: 18M grid cells
- CPU linear solver for pressure: geometric-algebraic multigrid (GAMG) of OpenFoam
- GPU linear solver for pressure: AMG preconditioned CG (AMGPCG) of Culises
- 200 SIMPLE iterations

<table>
<thead>
<tr>
<th>CPU cores Intel E5-2650</th>
<th>GPUs Nvidia K40</th>
<th>Linear solve time [s]</th>
<th>Total simulation time [s]</th>
<th>Speedup linear solver</th>
<th>Speedup total simulation</th>
<th>$f = \frac{\text{linear solve time}}{\text{total time}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>6819</td>
<td>13436</td>
<td>---</td>
<td>---</td>
<td>0.51</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1770</td>
<td>8383</td>
<td>3.85</td>
<td>1.60</td>
<td>---</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1264</td>
<td>7923</td>
<td>5.39</td>
<td>1.70</td>
<td>---</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>2912</td>
<td>6247</td>
<td>---</td>
<td>---</td>
<td>0.47</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1709</td>
<td>5094</td>
<td>1.70</td>
<td>1.23</td>
<td>---</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>1292</td>
<td>4695</td>
<td>2.25</td>
<td>1.33</td>
<td>---</td>
</tr>
</tbody>
</table>
Potential speedup for hybrid approach

Limited speedup acc. Amdahl's law:

\[ s = \frac{1}{(1 - f) + \frac{f}{a_{LS}}} \]

acceleration of linear solver on GPU:
- \( a_{LS} \to \infty, a_{MA} = 1.0 \)
- \( a_{LS} = 2.5, a_{MA} = 1.0 \)

\( a_{LS} \): Speedup linear solver
\( a_{MA} \): Speedup matrix assembly

fraction \( f = \frac{\text{CPU time spent in linear solver}}{\text{total CPU time}} \)

\( f(\text{steady-state run}) \ll f(\text{transient run}) \)

f: Solve linear system
1-f: Assembly of linear system
Summary hybrid approach

**Simulation tool e.g. OpenFOAM®**

- Preprocessing
- Discretization
- Linear solver
- Postprocessing

**Culises**

- Multigrid methods
- Krylov methods

**Advantage:**
- Universally applicable (coupled to simulation tool of choice)
- Full availability of existing flow models
- Easy/no validation needed

**Disadvantages:**
- Hybrid CPU-GPU produces overhead
- In case that solution of linear system not dominant (f<0.5)
  → Application speedup can be limited
Ongoing Development

Potential speedup for full GPU approach

\[ s = \frac{1}{f} \frac{1}{a_{LS}} + \frac{1}{1-f} \frac{1}{a_{MA}} \]

- \( a_{LS} \rightarrow \infty, a_{MA} = 1.0 \)
- \( a_{LS} = 2.5, a_{MA} = 1.0 \)
- \( a_{LS} = 2.5, a_{MA} = 2.0 \)

\( f \): Speedup linear solver
\( 1-f \): Speedup matrix assembly

\( f \) (steady-state run) \( \ll \) \( f \) (transient run)
Enhanced approach
Initially targeted flow solver

• Physical flow model:
  – Incompressible Navier-Stokes equations
  – Single-phase flow

• Numerical discretization method
  – Finite Volume (FV) method
    • Using unstructured mesh
    • Classical choice for
      – Flux (upwind)
      – Gradient evaluation,
      – Interpolation method, etc.
  – Pressure-velocity coupling using classical segregated approach:
    • SIMPLE method for steady-state flow
    • PISO method for transient flow

Profiling shows pre- & post-processing are negligible
→ mainly 2 parts dominate solution process:
(1) Assembly of linear systems (momentum and pressure correction)
(2) Solution of linear systems (Culises)
aeroFluidX
an extension of the hybrid approach

CPU flow solver
e.g. OpenFOAM®

aeroFluidX
GPU implementation

• Porting discretization of equations to GPU
  ➔ discretization module (Finite Volume) running on GPU
  ➔ Possibility of direct coupling to Culises
    ➔ Zero overhead from CPU-GPU-CPU memory transfer and matrix format conversion
    ➔ Solution of momentum equations also beneficial

• OpenFOAM® environment supported
  ➔ Enables plug-in solution for OpenFOAM® customers
  ➔ But communication with other input/output file formats possible
• CFD: simpleFoam solver (OpenFOAM® V2.2.2)
• GPU: aeroFluidX, that is not fully tuned/optimized yet!
• Fair comparison between OpenFOAM® and aeroFluidX
  – Linear solver:
    • Convergence criterion: satisfy same tolerance for norm of residual
    • Solver choice: select best available solver on CPU vs. best available linear solver on GPU
  – Discretization approach: use same methods for flux, gradient, interpolation, etc.
Validation: Re=400 (laminar), grid 250x250

CPU: Intel E5-2650 (all 8 cores)
GPU: Nvidia K40
4M grid cells (unstructured)
Running 100 SIMPLE steps with:
- OpenFOAM® (OF)
  - pressure: GAMG
  - Velocity: Gauss-Seidel
- OpenFOAM® (OFC)
  - Pressure: Culises AMGPCG (2.4x)
  - Velocity: Gauss-Seidel
- aeroFluidX (AFXC)
  - Pressure: Culises AMGPCG
  - Velocity: Culises Jacobi

Total speedup:
- OF (1x)
- OFC 1.62x
- AFXC 2.20x

Normalized computing time

all assembly = assembly of all linear systems (pressure and velocity)
all linear solve = solution of all linear systems (pressure and velocity)
aeroFluidX

NACA0012 airfoil flow

Validation: Re=2000 (laminar); angle of attack 0°; 40K grid cells

aeroFluidX
NACA0012 airfoil flow

- CPU: Intel E5-2650 (all 8 cores)
  GPU: Nvidia K40
- 4M grid cells (unstructured)
- Running 100 SIMPLE steps with:
  - OpenFOAM® (OF)
    - pressure: GAMG
    - Velocity: Gauss-Seidel
  - OpenFOAM® (OFC)
    - Pressure: Culises AMGPCG (1.5x)
    - Velocity: Gauss-Seidel
  - aeroFluidX (AFXC)
    - Pressure: Culises AMGPCG
    - Velocity: Culises Jacobi
- Total speedup:
  - OF (1x)
  - OFC 1.22x
  - AFXC 1.82x

![Normalized computing time graph]

- **all assembly** = assembly of all linear systems (pressure and velocity)
- **all linear solve** = solution of all linear systems (pressure and velocity)
**aeroFluidX**

**NACA0012 airfoil flow**

- CPU: Intel E5-2650 (all 8 cores)
  - GPU: Nvidia K40
- 8M grid cells (unstructured)
- Running 100 SIMPLE steps with:
  - OpenFOAM® (OF)
    - Pressure: GAMG
    - Velocity: Gauss-Seidel
  - OpenFOAM® (OFC)
    - Pressure: Culises AMGPCG (1.8x)
    - Velocity: Gauss-Seidel
  - aeroFluidX (AFXC)
    - Pressure: Culises AMGPCG
    - Velocity: Culises Jacobi
- Total speedup:
  - OF (1x)
  - OFC 1.34x
  - AFXC 2.03x

**Normalized computing time**

- all assembly = assembly of all linear systems (pressure and velocity)
- all linear solve = solution of all linear systems (pressure and velocity)
Conclusions

• **Culises** - hybrid approach for accelerated CFD applications (OpenFOAM)
  – **General applicability** for industrial cases including various existing flow models
  – Significant speedup (\(\geq 2x\)) of linear solver employing GPUs
  – **Moderate speedup** (\(\leq 1.6x\)) of total simulation
  – Culises V1.1 released: Commercial and academic licensing available
    Free testing & benchmarking opportunities at FluiDyna GPU-servers

• **aeroFluidX** - fully ported flow solver on GPU to harvest full GPU computing power
  – General applicability requires **rewrite of large large portion of existing code**
  – Steady-state, incompressible **unstructured multigrid** flow solver established & validated
  – Significant speedup (\(\geq 2x\)) of matrix assembly; **without full code tuning/optimization!**
  – **Enhanced speedup** (\(\geq 1.6x\)) of total simulation
Future development

• Enhancement of **aeroFluidX** solver:
  – RANS turbulence modeling (k-omega SST)
  – Multi-GPU capability (already started)
  – Support for moving/rotating geometries

• Further performance improvement in **Culisces**
  – Optimization of multi-GPU scaling and multi-node scaling
  – Classical coarsening for AMG
  – Additional smoothers