Design and Optimization of a Multi-stencil CFD Solver

Bahareh Davani · Ferran Martí Duran · Feng Liu
Aparna Chandramowlishwaran

August 2, 2017 — PADAL
CONTEXT: HiPer
(“High Performance Turbulent Flow Simulations”)
CONTEXT: HiPer
(“HIGH PERFORMANCE TURBULENT FLOW SIMULATIONS”)
Problem Formulation
GOVERNING EQUATIONS

- 3D Unsteady Reynolds Averaged Navier-Stokes (URANS) equations
- Dual time-stepping scheme
  - Pseudo-time marching — multi-stage Runge-Kutta scheme
  - Marched to a steady state in pseudo time
- Spatial discretization of the residual
  - 2nd order accurate
Read Mesh, Calculate volume & area
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

Update boundary condition
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

Calculate viscous flux, inviscid flux and artificial dissipation

Update boundary condition
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

Calculate viscous flux, inviscid flux and artificial dissipation

Update values at each cell

Update boundary condition
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

One stage of RK

Update values at each cell

Calculate viscous flux, inviscid flux and artificial dissipation

Update boundary condition
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

Update boundary condition

One stage of RK

Update values at each cell

Calculate viscous flux, inviscid flux and artificial dissipation
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

Calculate viscous flux, inviscid flux and artificial dissipation

Update boundary condition

One stage of RK

Update values at each cell

After the 5th stage

Calculate the residual

Calculate the residual
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

One stage of RK

Update values at each cell

Calculate the residual

Calculate viscous flux, inviscid flux and artificial dissipation

Update boundary condition

After the 5th stage

Solution not converged
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

One stage of RK

Update values at each cell

Calculate viscous flux, inviscid flux and artificial dissipation

Update boundary condition

After the 5th stage

Calculate the residual
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

One stage of RK

Update values at each cell

Calculate viscous flux, inviscid flux and artificial dissipation

Update boundary condition

After the 5th stage

Calculate the residual

Solution converged

Collect results
Stencil Patterns

- **Cell-centered stencils**
  - Most well-studied in literature

- **Vertex-centered stencils**
  - More complex memory access pattern
  - More memory-bound than cell-centered stencils
Stencil Patterns

- **Cell-centered stencils**
  - Most well-studied in literature

- **Vertex-centered stencils**
  - More complex memory access pattern
  - More memory-bound than cell-centered stencils
Stencil Patterns

- Cell-centered stencils
  - Most well-studied in literature
- Vertex-centered stencils
  - More complex memory access pattern
  - More memory-bound than cell-centered stencils
Stencil Patterns

- **Cell-centered stencils**
  - Most well-studied in literature

- **Vertex-centered stencils**
  - More complex memory access pattern
  - More memory-bound than cell-centered stencils
Stencil Patterns

- Cell-centered stencils
  - Most well-studied in literature

- Vertex-centered stencils
  - More complex memory access pattern
  - More memory-bound than cell-centered stencils
Stencil Patterns

- Cell-centered stencils
  - Most well-studied in literature

- Vertex-centered stencils
  - More complex memory access pattern
  - More memory-bound than cell-centered stencils
Stencil Patterns

- Cell-centered stencils
  - Most well-studied in literature
- Vertex-centered stencils
  - More complex memory access pattern
  - More memory-bound than cell-centered stencils
Stencil Patterns

- Cell-centered stencils
  - Most well-studied in literature

- Vertex-centered stencils
  - More complex memory access pattern
  - More memory-bound than cell-centered stencils
OPTIMIZATIONS

- Single-core, manually coded & tuned
  - *Low-level*: SIMD vectorization (x86), strength reduction
  - *Data*: Structure reorg. (transpose or “SOA”)
  - *Traffic*: Intra-stencil and inter-stencil fusion, cache blocking
- NUMA-aware OpenMP parallelization
OPTIMIZATIONS

- Single-core, manually coded & tuned
  - Low-level: SIMD vectorization (x86), strength reduction
  - Data: Structure reorg. (transpose or “SOA”)
  - Traffic: Intra-stencil and inter-stencil fusion, cache blocking
- NUMA-aware OpenMP parallelization
Single- and Multi-core Optimizations (Step flow with 2 million cells)

~349

~277x

~19x

~ 9x

Number of threads

Number of threads

Optimization
+Blocking
+SIMD
+SIMD Code Struct
+NUMA
+Parallelism
+Fusion
+Strength Reduction

Optimization
+Blocking
+SIMD
+SIMD Code Struct
+NUMA
+Parallelism
+Fusion
+Strength Reduction

Abu Dhabi

Haswell

Speedup

Multiple sockets

Hyperthreading

Single- and Multi-core Optimizations (Step flow with 2 million cells)
Single- and Multi-core Optimizations
(Step flow with 2 million cells)
Intra-stencil fusion

Gradients of Velocity

Viscous Flux

Add incoming flux in i direction

Add incoming flux in j direction

Add incoming flux in k direction

Inter-stencil fusion
Improving **locality** and **parallelism** requires trading off **redundant work**.
Single- and Multi-core Optimizations
(Step flow with 2 million cells)
Read Mesh, Calculate volume & area

Start iteration (5-stage Runge Kutta)

One stage of RK

Update values at each cell

Calculate residuals

Calculate viscous flux, inviscid flux and artificial dissipation

Update boundary condition

After the 5th stage

Solution not converged

Solution converged

Collect results
Further improving **locality** requires trading off **accuracy**.
The preceding optimizations were manually coded. Can such CFD solvers can be expressed in stencil DSL’s?
The preceding optimizations were manually coded. Can such CFD solvers can be expressed in stencil DSL’s?

Yes! 1 month effort in Halide.

Can stencil DSL’s deliver a sufficient combination of optimizations to compete with a hand-tuned code?
<table>
<thead>
<tr>
<th>Optimization</th>
<th>Haswell</th>
<th>Abu Dhabi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-tuned</td>
<td>4x</td>
<td>3.3x</td>
</tr>
<tr>
<td>Halide</td>
<td>2x</td>
<td>1.23x</td>
</tr>
</tbody>
</table>

This gap is due to **strength reduction** and **inter-stencil fusion** in the hand-tuned code.
<table>
<thead>
<tr>
<th></th>
<th>Haswell</th>
<th>Abu Dhabi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-tuned</td>
<td>4x</td>
<td>3.3x</td>
</tr>
<tr>
<td>Halide</td>
<td>2x</td>
<td>1.23x</td>
</tr>
<tr>
<td>Optimization</td>
<td>4x</td>
<td>2x</td>
</tr>
<tr>
<td>+Parallelization</td>
<td>21.8x</td>
<td>6x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37.8x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5x</td>
</tr>
</tbody>
</table>

This gap is partly due to **NUMA-aware parallelization** in the hand-tuned code. (Halide is currently not NUMA-aware)
<table>
<thead>
<tr>
<th>Feature</th>
<th>Haswell Hand-tuned</th>
<th>Haswell Halide</th>
<th>Abu Dhabi Hand-tuned</th>
<th>Abu Dhabi Halide</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization</td>
<td>4x</td>
<td>2x</td>
<td>3.3x</td>
<td>1.23x</td>
</tr>
<tr>
<td>+Parallelization</td>
<td>21.8x</td>
<td>6x</td>
<td>37.8x</td>
<td>5x</td>
</tr>
<tr>
<td>+Vectorization</td>
<td>2.8x</td>
<td>1.1x</td>
<td>1.65x</td>
<td>1x</td>
</tr>
</tbody>
</table>
Can stencil DSL’s deliver a sufficient combination of optimizations to compete with a hand-tuned code?

Not yet! But, there is hope.
CONCLUSIONS

Improving **locality** and **parallelism** requires trading off **redundant work** and **accuracy**.

CFD solvers can be expressed in stencil DSL’s with minimal effort.

**Limitations**

- Finding the optimal schedule for performance is non-trivial.
- Most DSL’s are only optimized for cell-centered stencils.
- Does not support sufficient combination of optimizations to compete with hand-tuned code yet.