Overview of the Dagstuhl Seminar on “Advanced Stencil Code Engineering”

Georg Hager
Erlangen Regional Computing Center (RRZE)

Seminar on Advanced Stencil Code Engineering
April 13-17, 2015, Schloss Dagstuhl, Wadern, Germany
### Monday

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>08:45</td>
<td>Lengauer</td>
<td>Welcome, introductions, information for the week</td>
</tr>
<tr>
<td>09:40</td>
<td>Rüde</td>
<td>From stencils to elliptic PDE solvers</td>
</tr>
<tr>
<td>10:50</td>
<td>Engwer</td>
<td>Stencil-like operations on unstructured meshes</td>
</tr>
<tr>
<td>11:30</td>
<td>Hager</td>
<td>Optimization opportunities of stencils codes via analytic performance modeling</td>
</tr>
<tr>
<td>14:00</td>
<td>Lengauer</td>
<td>SPPEXA and ExaStencils</td>
</tr>
<tr>
<td>14:40</td>
<td>Cardiel</td>
<td>LFA for multigrid on semi-structured meshes</td>
</tr>
<tr>
<td>16:00</td>
<td>Friedhoff</td>
<td>Predicting the numerical performance of methods for evolutionary problems</td>
</tr>
<tr>
<td>16:40</td>
<td>Yang</td>
<td>An extension of hypre’s structured and semi-structured matrix classes</td>
</tr>
<tr>
<td>17:20</td>
<td>Kelly</td>
<td>The PyOP2 abstraction</td>
</tr>
</tbody>
</table>

### Tuesday

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00</td>
<td>Kuszmaul</td>
<td>The Pochoir stencil compiler</td>
</tr>
<tr>
<td>09:40</td>
<td>Franchetti</td>
<td>Formals Synthesis of Computational Kernels</td>
</tr>
<tr>
<td>10:50</td>
<td>ExaStencils</td>
<td>ExaSlang and the ExaStencils code generator</td>
</tr>
<tr>
<td>14:00</td>
<td>Grebhahn</td>
<td>Variability management in ExaStencils</td>
</tr>
<tr>
<td>14:40</td>
<td>Solar-Lezama</td>
<td>From general-purpose to stencil DSL code</td>
</tr>
<tr>
<td>16:00</td>
<td>Osuna</td>
<td>The stencil DSL STELLA</td>
</tr>
<tr>
<td>16:40</td>
<td>Bianco</td>
<td>Designing GridTools</td>
</tr>
<tr>
<td>17:20</td>
<td>Vanroose</td>
<td>Redesign of preconditioned Krylov methods around stencil compilers</td>
</tr>
</tbody>
</table>
## Program continued

### Wednesday

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00</td>
<td>Bondhugula</td>
<td><em>PolyMage: High-performance compilation for heterogeneous stencils</em></td>
</tr>
<tr>
<td>09:40</td>
<td>Sadayappan</td>
<td>On the characterization of the data movement complexity of algorithms</td>
</tr>
<tr>
<td>10:50</td>
<td>Rittich</td>
<td>The mathematics of ExaStencils</td>
</tr>
<tr>
<td>11:30</td>
<td>Pflüger</td>
<td>Stencils for hierarchical bases</td>
</tr>
</tbody>
</table>

### Thursday

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00</td>
<td>Pouchet</td>
<td>Mapping stencils to FPGAs and synthesizable accelerators</td>
</tr>
<tr>
<td>09:40</td>
<td>Grosser</td>
<td>The stencil optimization framework MODESTO</td>
</tr>
<tr>
<td>10:50</td>
<td>Palamadai</td>
<td>Autotuning divide-and-conquer stencil computations</td>
</tr>
<tr>
<td>11:30</td>
<td>Wittum</td>
<td>Things you can do with stencils</td>
</tr>
<tr>
<td>14:00</td>
<td>Saday</td>
<td><strong>Optimization of higher-order stencils</strong></td>
</tr>
<tr>
<td>14:40</td>
<td>Haase</td>
<td>DSL for stencils in non-Newtonian fluids simulation?</td>
</tr>
</tbody>
</table>
B. Kuszmaul: The Pochoir Stencil Compiler

Dagstuhl April 14 2015

The Pochoir Stencil Compiler

Yuan Tang (Fudan) Rezaul Alam Chowdhury (Stony Brook)
Bradley C. Kuszmaul (MIT CSAIL) Chi-Keung Luk (Intel)
Charles E. Leiserson (MIT CSAIL)

© 2011 Charles E. Leiserson
Pochoir

**Pochoir Stencil Compiler**

- Domain-specific compiler programmed in Haskell that compiles a stencil language embedded in C++, a traditionally difficult language in which to embed a separately compiled domain-specific language.
- Employs a novel cache-oblivious algorithm for arbitrary d-dimensional grids which is parallelized using Intel Cilk Plus.
- Makes it straightforward to code arbitrary periodic and nonperiodic boundary conditions, including Neumann and Dirichlet conditions.
- Implements a variety of stencil-specific optimizations.
- The stencil specification can be executed and debugged without the Pochoir compiler.
Pochoir

**Benchmarking Platform**

- Intel C++ version 12.0.0 compiler with Intel Cilk Plus.
- 12-core Intel Core i7 (Nehalem) machine, where each core has a private 32-KB L1-data-cache, a private 256-KB L2-cache, and a shared 12-MB L3-cache.
Pochoir

2D Heat Equation

5-point stencil on a torus

Pochoir vs. Parallel Loop vs. Serial Loop
19-point stencil on a nonperiodic domain

Measured limit for best RRZE variant (no temp. blocking)

Pochoir vs. Parallel Loop vs. Serial Loop
Pochoir

**TWO-PHASE COMPILATION STRATEGY**

**Phase 1 goal:**
Check functional correctness

**Phase 2 goal:**
Maximize performance

© 2011 Charles E. Leiserson

*The Pochoir Stencil Compiler*
Raising the Arithmetic Intensity of Krylov solvers

1 Applied Mathematics, University of Antwerp, Belgium
2 Future Technology Group, Berkeley Lab, USA
3 Intel ExaScience Lab, Belgium
4 USI, Lugano, CH

B. Reps¹, P. Ghysels², S. Donfack⁴, O. Schenk⁴, W. Vanroose¹,³

www.exa2ct.eu
Pluto for CA Krylov

Arithmetic intensity of \( k \) dependent SpMVs

<table>
<thead>
<tr>
<th></th>
<th>1 SpMV</th>
<th>( k ) SpMVs</th>
<th>( k ) SpMVs in place</th>
</tr>
</thead>
<tbody>
<tr>
<td>flops</td>
<td>( 2n_{nz} )</td>
<td>( 2k \cdot n_{nz} )</td>
<td>( 2k \cdot n_{nz} )</td>
</tr>
<tr>
<td>words moved</td>
<td>( n_{nz} + 2n )</td>
<td>( kn_{nz} + 2kn )</td>
<td>( n_{nz} + 2n )</td>
</tr>
<tr>
<td>q</td>
<td>2</td>
<td>2</td>
<td>( 2k )</td>
</tr>
</tbody>
</table>

J. Demmel, CS 294-76 on Communication-Avoiding algorithms

Pluto for CA Krylov

PLUTO (Temporal blocking)

Increase the arithmetic intensity based on the underlying stencils.

- Divide the domain in tiles which fit in the cache.
- Reuse the data in the tiles \( k \) times.
- Use compiler auto-vectorization capabilities within each tiles.

- **Increased** parallelism and temporal data reuse.
- Inner-tiles suitable for external spatial blocking tools e.g PATUS.
- Tiles representation suitable for most efficient scheduling strategy e.g TBB.
Pluto for CA Krylov

Increasing the arithmetic intensity with a stencil compiler (Pluto)

Dual socket Intel Sandy Bridge, 16 × 2 threads

Intel Xeon Phi, 61 × 4 threads
Stencil DSL for image processing

PolyMage: High-Performance Compilation for Heterogeneous Stencils

Uday Bondhugula

(with Ravi Teja Mullapudi, Vinay Vasista)

Department of Computer Science and Automation
Indian Institute of Science
Bangalore, India

Apr 15, 2015
Stencil DSL for image processing

Graphs of interconnected processing stages

Figure: Harris corner detection
Stencil DSL for image processing

Example: Pyramid Blending pipeline

Image courtesy: Kyros Kutulakos
Stencil DSL for image processing

**Harris Corner Detection**

\[\begin{align*}
R, c & \text{ Parameter(int)} & Parameter(int) \\
I & \text{ Image(Float)} & [2\times2, 0\times0] \\
x, y & \text{ Variable()} & \text{ Variable()} \\
\text{row, col} & \text{ Interval(0, index, 1)} & \text{ Interval(0, index, 1)} \\
c & \text{ Condition}(x, y, \geq 0) & \text{ Condition}(x, y, \leq 0) & \text{ Condition}(y, x, \geq 0) & \text{ Condition}(y, x, \leq 0) \\
cb & \text{ Condition}(x, y, \geq 0, 2) & \text{ Condition}(x, y, \leq 0, 2) & \text{ Condition}(y, x, \geq 0, 2) & \text{ Condition}(y, x, \leq 0, 2) \\
\end{align*}\]

\[\text{Iy} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{Iy}.\text{defn} = \{ \text{ Case}(c, \text{Stencil}(I_y, y), 1.0/12, \{[-1, -2, -1], [0, 0, 0], [1, 2, 1]\}) \}\]

\[\text{Ix} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{Ix}.\text{defn} = \{ \text{ Case}(c, \text{Stencil}(I_x, x), 1.0/12, \{[-1, 0, 1], [-2, 0, 2], [-1, 0, 2]\}) \}\]

\[\text{Ixx} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{Ixx}.\text{defn} = \{ \text{ Case}(c, \text{Ix}(x, y) + \text{Ix}(x, y)) \}\]

\[\text{Iyy} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{Iyy}.\text{defn} = \{ \text{ Case}(c, \text{Iy}(x, y) + \text{Iy}(x, y)) \}\]

\[\text{Szxy} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{Szxy}.\text{defn} = \{ \text{ Case}(c, \text{Ix}(x, y) + \text{Iy}(x, y)) \}\]

\[\text{Sxy} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{Sxy}.\text{defn} = \{ \text{ Case}(c, \text{Iy}(x, y) + \text{Ix}(x, y)) \}\]

\[\text{det} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{det}.\text{defn} = \{ \text{ Case}(c, \text{Ix}(x, y) + \text{Iy}(x, y)) \}\]

\[\text{trace} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{trace}.\text{defn} = \{ \text{ Case}(c, 2.0 \times \text{det}(x, y) + \text{trace}(x, y)) \}\]

\[\text{harris} = \text{ Function}(\text{varDom} = \{(x, y), \text{[row, col]}\}, \text{Float})\]
\[\text{harris}.\text{defn} = \{ \text{ Case}(c, \text{det}(x, y) - \text{trace}(x, y)^2) \}\]
Stencil DSL for image processing

Scheduling Criteria

Overlap tiling

Parallelism  Locality  Storage

Re-computation

Uday Bondhugula, Indian Institute of Science  Dagstuhl seminar, Apr 12-17, 2015
Stencil DSL for image processing

Effectiveness of Schedule Transformations

Speedup of grouped and tiled implementations over naively parallelized and vectorized ones

16 threads and vectorization enabled
- PolyMage optimized code exhibits better scaling, better vectorization efficiency due to better locality

Uday Bondhugula, Indian Institute of Science  Dagstuhl seminar, Apr 12-17, 2015
High-order stencils optimization

Optimization of High-Order Stencils*

Kevin Stock  Ohio State University
Martin Kong  Ohio State University
Louis-Noel Pouchet  Ohio State University
Fabrice Rastello  INRIA, Grenoble
J. (Ram) Ramanujam  Louisiana State University
Saday Sadayappan  Ohio State University

* Funded in part by NSF and DOE
High-order stencils optimization

Optimizing High-Order Stencils

- High-order stencils: weighted averaging over wider neighborhood
- “Box” wxw stencil => $O(w^2)$ flops per site ($w=2*k+1$)
- Operational intensity: $O(w^2)$

```
for (i=k; i<n-k; i++)
  for (j=k; j<n-k; j++)
    // ii, jj loops fully unrolled
    for (ii=-k; ii<=k; ii++)
      for (jj=-k; jj<=k; jj++)
        B[i][j] += W[k+ii][k+jj]*A[i+ii][j+jj];
```
High-order stencils optimization

High-Order Stencils: Baseline Performance

for (i=k; i<n-k; i++)
for (j=k; j<n-k; j++)
// ii,jj loops fully unrolled
for (ii=-k; ii<=k; ii++)
for (jj=-k; jj<=k; jj++)
\[ B[i][j] \leftarrow W[k+ii][k+jj]*A[i+ii][j+jj]; \]

- Multi-core performance improves from 3x3 to 5x5 stencil but drops for higher order stencils
- Problem:
  - Not memory-bw bound
  - But too many register loads due to insufficient registers to get reuse
High-order stencils optimization

Reordering Stencil Operations

- wwx “box” stencil computation over MxN domain viewed as a total of \(w^2MN\) edges
- Standard stencil execution groups all edges incoming into a site for consecutive execution
- Regrouping of edges forms a different “stencil” pattern, with different #loads/stores and register pressure
High-order stencils optimization

Gather-Scatter

- 1 reads from *IN*
- *w* – 1 reads from *OUT*
- *w* write to *OUT*
- *w* + 1 registers

Scatter-Gather

- *w* reads from *IN*
- 0 reads from *OUT*
- 1 write to *OUT*
- *w* + 1 registers
High-order stencils optimization

Stencils on Vector-SIMD Processors

for (i=0; i<H; ++i)
for (j=0; j<W; ++j)
z[i][j] += y[i][j] + y[i][j+1];

- Fundamental source of inefficiency with stencil codes on current short-vector SIMD ISAs (e.g. SSE, AVX ...)
  - Concurrent operations on contiguous elements
  - Each data element is reused in different "slots" of vector register
  - Redundant loads or shuffle ops needed

- Data Layout transformation based on matching computational characteristics of stencils to vector-SIMD architecture characteristics

Inefficiency: Each element of \( b \) is loaded twice
High-order stencils optimization

Performance: Stencil Rate

Benchmark

MStencil/s

REF-PAR  OPT-PAR  DLTOPT-PAR
REF-SEQ  OPT-SEQ  DLTOPT-SEQ

2d-d-1  2d-d-2  2d-d-3  2d-d-4  2d-d-5  2d-d-6  2d-f-1  2d-f-2  2d-f-3  2d-f-4  2d-f-5  2d-f-6  3d-d-1  3d-d-2  3d-f-1  3d-f-2  4d-f-1
High-order stencils optimization