Advanced Stencil-Code Engineering (ExaStencils)

Christian Lengauer

3rd Workshop on Extreme-Scale Programming Tools
New Orleans, 17.11.2014
Two Alternative Approaches in SPPEXA

- **The evolutionary approach**
  
  \[
  \begin{array}{c}
  \text{Fortran} \\
  \text{C} \\
  \text{Java}
  \end{array}
  \begin{array}{c}
  + \\
  \text{MPI} \\
  \text{OpenMP} \\
  \text{Threads}
  \end{array}
  \]

- **The revolutionary approach**

  linear transformations
  linear algebra
  stencil codes

  domain-specific refinement and optimization

  HPC cluster
  Manycores
  GPGPUs
  FPGAs

ExaStencils, ESPT-SC 2014, Christian Lengauer
Why is this Revolutionary?

- **No general-purpose programming language**
  - Different refinement levels have their own domain-specific language
  - *Exploitation of domain knowledge at all levels for refinement*
    - About input data
    - About the algorithm
    - About the execution platform
  - *Exploitation of common properties of programs*
    - Programs are not individuals but members of a family, a "product line”
    - A product line specifies variabilities (so-called features)
    - Common properties and individual variations are stated explicitly and precisely
    - The "programming" of a product is done by selection options (and nothing else!)
  - *Still – the promise: full automation*
    - The target code is being "weaved" automatically, optimized for the features selected
    - The optimization exploits knowledge about the specific feature combination
A new, tool-assisted, domain-specific codesign approach for stencil codes
Work Flow of ExaStencils
Domain: Multigrid Stencil Codes

The Multigrid V-cycle

Smooth

Finest Grid

Restriction

Fewer Dofs

First Coarse Grid

Prolongation

© Robert Falgout
Hierarchical Hybrid Grids
Algorithmic Engineering

Variabilities
- Discretization method
- Grid transfer method
- Cycling strategy
- Smoother

Variants must reduce
- Convergence rate (platform-independent)
- Execution time (platform-dependent)

Current activities
- A-priori prediction of the convergence rate by local Fourier analysis (LFA)
- Extend LFA techniques to block-smoothers and aggressive coarsening
**Domain-Specific Representation**

- **Abstract problem formulation**
- **Concrete solver implementation**

---

**Layer 1:**
Continuous Domain & Continuous Model

**Layer 2:**
Discrete Domain & Discrete Model

**Layer 3:**
Algorithmic Components & Parameters

**Layer 4:**
Complete Program Specification

---

**Current status**

- Educated choice of Scala as the host language
- Preliminary code generator for proof of concept finalized
- Serious code generator for 80% of Layer 4 shows exascale potential
Domain-Specific Modelling and Optimization

**Challenges**
- Draft the software product line: identify the variabilities
- Configuration options can interact in subtle ways (feature interaction)
- Which combination of options gives the best performance?

**Current status**
- Adoption of techniques of automated software configuration
- Design of a variability model for the Highly Scalable Multigrid Solver (HSMGS)
- First experiments with a machine learning approach to identify efficient configurations
- New: not only binary but also numerically parameterized options
- Measurements of 10.2% of all variants → prediction accuracy of 89% on ave.
- Still, so far, no domain-specific knowledge exploited!
Highly Scalable Multigrid Solver (HSMGS)

pre-smoothing
[0..6]
3

post-smoothing
[0..6]
3

coarse grid solver

smoother

IP_CG  RED_AMG  IP_AMG  Jac  GS  GSAC  RBGS  RBGSAC  BS

sum (pre-smoothing, post-smoothing) > 0

Legend:
IP_CG = In-Place Conjugate Gradient
IP_AMG = In-Place Algebraic multigrid
RED_AMG = Algebraic multigrid with data reduction
GSAC = Gauss-Seidel with additional communication
RBGSAC = Red-Black Gauss-Seidel with additional communication
Jac = Jacobi
GS = Gauss-Seidel
RBGS = Red-Black Gauss-Seidel
BS = Block-Smoother
## Preliminary Code Generator

<table>
<thead>
<tr>
<th>Variability</th>
<th>Layer</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computational domain</strong></td>
<td>DSL 1</td>
<td>UnitSquare, UnitCube</td>
</tr>
<tr>
<td><strong>Operator</strong></td>
<td>DSL 1</td>
<td>Laplacian, ComplexDiffusion</td>
</tr>
<tr>
<td><strong>Boundary conditions</strong></td>
<td>DSL 1</td>
<td>Dirichlet, Neumann</td>
</tr>
<tr>
<td>Location of grid points</td>
<td>DSL 2</td>
<td>node-based, cell-centered</td>
</tr>
<tr>
<td>Discretization</td>
<td>DSL 2</td>
<td>finite differences, finite volumes</td>
</tr>
<tr>
<td>Data type</td>
<td>DSL 2</td>
<td>single/double accuracy, complex numbers</td>
</tr>
<tr>
<td>Multigrid smoother</td>
<td>DSL 3</td>
<td>$\omega$-Jacobi, $\omega$-Gauss-Seidel, red-black variants</td>
</tr>
<tr>
<td>Multigrid inter-grid transfer</td>
<td>DSL 3</td>
<td>constant and linear interpolation and restriction</td>
</tr>
<tr>
<td>Multigrid coarsening</td>
<td>DSL 3</td>
<td>direct (re-discretization)</td>
</tr>
<tr>
<td>Multigrid parameters</td>
<td>DSL 3</td>
<td>various</td>
</tr>
<tr>
<td>Platform</td>
<td>Hardware</td>
<td>CPU, GPU</td>
</tr>
<tr>
<td>Parallelization</td>
<td>Hardware</td>
<td>serial, OpenMP</td>
</tr>
</tbody>
</table>

**Novelties:**
- Variant-driven code generation
- Wide spectrum of stencil codes
What Makes a Domain Suitable for the Radical Approach?

- **Size**
  - Considerably smaller than by contemporary expectations

- **Theoretical basis:**
  - Algebra
  - Conditional equations

- **Significance**
  - Stable abstract view
  - Stable, sustained user community

- **Examples**
  - FFTW: the fastest Fourier transform in the West
  - Spiral: discrete linear transforms
  - DBMSs: relational query optimization
  - cpp: Linux operating system configuration
  - ExaStencils: Multigrid stencil codes
Thanks for your Interest in the World of Stencils

(a) Laplacian
3D 7-point stencil, scalar → scalar

(b) Divergence
3D 6-point stencil, vector → scalar

(c) Gradient
3D 6-point stencil, scalar → vector

(d) Hyperthermia
3D 7-point stencil, scalar + 9 coefficients → scalar

(e) 6th order Laplacian
3D 19-point stencil, scalar → scalar

(f) Tricubic interpolation
3D 64-point stencil, scalar → vector

(g) Wave
3D 13-point stencil, scalar → scalar depending on 2 time steps

(h) Edge detection / Game of Life
2D 9-point stencil, scalar → scalar

© Matthias Christen