TORSTEN HOEFLER

MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures

with support of Tobias Gysi, Tobias Grosser @ SPCL presented at Guangzhou, China, Sept. 2016
ETH, CS, Systems Group, SPCL

- ETH Zurich – top university in central Europe
  - Shanghai ranking ’15 (Computer Science): #17, best outside North America
  - 16 departments, 1.62 Bn $ federal budget
  - CSCS is part of ETH (houses Europe’s fastest HP machine)

- Computer Science department
  - 28 tenure-track faculty, 1k students

- Systems group (7 professors)
  - Focused on systems research of all kinds (data management, OS, ...)

- SPCL focusses on performance/data/HPC
  - 1 faculty
  - 3 postdocs
  - 8 PhD students (+2 external)
  - 15+ BSc and MSc students
  - [http://spcl.inf.ethz.ch](http://spcl.inf.ethz.ch)
  - Twitter: @spcl_eth
Scientific **Performance** Engineering

1) Observe

2) Model

\[ E = mc^2 \]

3) Understand

4) Build

More at keynote at HPC China next month in Xi’an!
Stencil computations (oh no, another stencil talk)

Motivation:
- Important algorithmic motif (e.g., finite difference method)

Definition:
- Element-wise computation on a regular grid using a fixed neighborhood
- Typically working on multiple input fields and writing a single output field

\[
\text{lap}(i,j) = -4.0 \times \text{in}(i,j) + \text{in}(i-1,j) + \text{in}(i+1,j) + \text{in}(i,j-1) + \text{in}(i,j+1)
\]

due to their low arithmetic intensity, stencil computations are typically heavily memory bandwidth limited!
How to tune such stencils (most other stencil talks)

- LOTS of related work!
  - Compiler-based (e.g., Polyhedral such as PLUTO [1])
  - Auto-tuning (e.g., PATUS [2])
  - Manual model-based tuning (e.g., Datta et al. [3])
  - ... essentially every micro-benchmark or tutorial, e.g.:

- Common features
  - Vectorization tricks (data layout)
  - Advanced communication (e.g., MPI neighbor colls)
  - Tiling in time, space (diamond etc.)
  - Pipelining

- Much of that work DOES NOT compose well with practical complex stencil programs

---

[1]: Uday Bondhugula, A. Hartono, J. Ramanujan, P. Sadayappan. A Practical Automatic Polyhedral Parallelizer and Locality Optimizer, PLDI'08
[3]: Kaushik Datta, et al., Optimization and Performance Modeling of Stencil Computations on Modern Microprocessors, SIAM review
What is a “complex stencil program”? (this stencil talk)

E.g., the COSMO weather code
- is a regional climate model used by 7 national weather services
- contains hundreds of different complex stencils

Modeling stencils formally:
- Represent stencils as DAGs
  - Model stencil as nodes, data dependencies as edges

What is a “complex stencil program”? (this stencil talk)

Simplified horizontal diffusion example

\[ a \oplus b = \{ a' + b' \mid a' \in a, b' \in b \} \]
Data-locality Transformations

- Consider the horizontal diffusion lap-fli-out dependency chain (i-dimension)

Loop Tiling & Loop Fusion

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS’15
How to Deal with Data Dependencies?

- Consider the horizontal diffusion lap-fli-out dependency chain (i-dimension)

**Halo Exchange Parallel (hp):**
- Update tiles in parallel
- Perform halo exchange communication

**Pros and Cons:**
- Avoid redundant computation
- At the cost of additional synchronization

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS’15
How to Deal with Data Dependencies?

- Consider the horizontal diffusion lap-fli-out dependency chain (i-dimension)

Halo Exchange Sequential (hs):
- Update tiles sequentially
- Innermost loop updates tile-by-tile

Pros and Cons:
- Avoid redundant computation
- At cost of being sequential
How to Deal with Data Dependencies?

- Consider the horizontal diffusion lap-fli-out dependency chain (i-dimension)

Computation on-the-fly (of):
- Compute all dependencies on-the-fly
- Overlapped tiling

Pros and Cons:
- Avoid synchronization
- At the cost of redundant computation

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS’15
Hierarchical Tiling

- By tiling the domain repeatedly we target multiple memory hierarchy levels

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS'15
Case Study: STELLA (STEncil Loop LAnguage)

- STELLA is a C++ stencil DS(e)L of COSMO’s dynamical core (50k LOC, 60% RT)

```cpp
// define stencil functors
struct Lap { ... };  
struct Fli { ... };  
...
// stencil assembly
Stencil stencil;
StencilCompiler::Build(
    stencil,
    pack_parameters(...),
    define_temporaries(
        StencilBuffer<lap, double>(),
        StencilBuffer<fli, double>(),
        ...),
    define_loops(
        define_sweep(
            StencilStage<Lap, IJRange<-1,1,-1,1> >(),
            StencilStage<Fli, IJRange<-1,0,0,0> >(),
            ...
        )),
    // stencil execution
    stencil.Apply();
```

using C++ template metaprogramming:

- STELLA defines a virtual tiling hierarchy that facilitates platform independent code generation
# Tiling Hierarchy of STELLA’s GPU-Backend

<table>
<thead>
<tr>
<th>DSL</th>
<th>Tile Size</th>
<th>Strategy</th>
<th>Memory</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>sweep</td>
<td>$1 \times 1 \times 1$</td>
<td>halo exchange parallel</td>
<td>registers</td>
<td>scratchpad</td>
</tr>
<tr>
<td>sweep</td>
<td>$\infty \times \infty \times 1$</td>
<td>halo exchange sequential</td>
<td>registers</td>
<td>registers</td>
</tr>
<tr>
<td>loop</td>
<td>$64 \times 4 \times 64$</td>
<td>computation on-the-fly</td>
<td>GDDR</td>
<td>-</td>
</tr>
<tr>
<td>stencil</td>
<td>$\infty \times \infty \times \infty$</td>
<td>computation on-the-fly</td>
<td>GDDR</td>
<td>-</td>
</tr>
</tbody>
</table>

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS’15
Stencil Program Algebra

- Map stencils to the tiling hierarchy using a bracket expression

- Enumerate the stencil execution orders that respect the dependencies

- Enumerate implementation variants by adding/removing brackets

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS'15
Our model considers peak computation and communication throughputs.

- **Target Machine**:
  - Core 1: (30 Gflop)
  - Core 2: (30 Gflop)
  - Core 3: (30 Gflop)
  - Cache: (256 kB)
  - DDR: (8 GB)

- **Machine Model**:
  - \( C = 90 \text{ Gflops} \)
  - \( V^1 = 300 \text{ GB/s} \)
  - \( L^1 = 50 \text{ GB/s} \)
  - \( M^1 = 256 \text{ kB} \)
  - \( V^0 = 10 \text{ GB/s} \)
  - \( L^0 = 0 \text{ GB/s} \)
  - \( M^0 = 8 \text{ GB} \)

Lateral and vertical communication refer to communication within one respectively between different tiling hierarchy levels.
**Stencil Performance Model - Overview**

- Given a stencil $s$ given and the amount of computation $c_s$
  
  $t_s = \frac{c_s}{C}$

- Given a group $g$ and the vertical and lateral communication $v_c$ and $l_c^1, ..., l_c^m$
  
  $t_g = \sum_{c \in g.\text{child}} \max(t_c, v_c/V^m, l_c^1/L^1, ..., l_c^m/L^m)$

---

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS’15
Stencil Performance Model - Affine Sets and Maps

- The stencil program analysis is based on (quasi-) affine sets and maps
  \[ S = \{ \vec{i} | \vec{i} \in \mathbb{Z}^n \land (0, \ldots, 0) < \vec{i} < (10, \ldots, 10) \} \]
  \[ M = \{ \vec{i} \rightarrow \vec{j} | \vec{i} \in \mathbb{Z}^n, \vec{j} \in \mathbb{Z}^n \land \vec{j} = 2 \cdot \vec{i} \} \]
- For example, data dependencies can be expressed using named maps
  \[ D_{fli} = \{ (fli, \vec{i}) \rightarrow (lap, \vec{i} + \vec{j}) | \vec{i} \in \mathbb{Z}^2, \vec{j} \in \{(0,0), (1,0)\} \} \]

\[
D = D_{lap} \cup D_{fli} \cup D_{flj} \cup D_{out}
\]
\[
E = D^+\left(\{(out, \vec{0})\}\right)
\]

apply the out origin vector to the transitive closure of all dependencies
Stencil Performance Model - Tiling Transformations

- Define a tiling using a map that associates stencil evaluations to tile ids

\[ T_{out} = \{(\text{out}, (i_0, i_1)) \rightarrow ([i_0/2], [i_1/2])\} \]
Stencil Performance Model – Comp & Comm

- Count floating point operations necessary to update tile (0,0)

\[ c_{out} = |T_{out} \cap ran \{(0,0)\}| \cdot \# flops \]

- Count the number of loads necessary to update tile (0,0)

\[ l_{out} = |(T_{out} \circ D_{out}^{-1}) \cap ran \{(0,0)\}| \]
Analytic Stencil Program Optimization

- Put it all together (stencil algebra, performance model, stencil analysis)
  1. Optimize the stencil execution order (brute force search)
  2. Optimize the stencil grouping (dynamic programming / brute force search)

\[
\min_{x \in I} t(x)
\]

subject to \( m(x) \leq M \)

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS'15
Evaluation

**CPU Experiments (i5-3330):**

<table>
<thead>
<tr>
<th></th>
<th>no fusion</th>
<th>hand-tuned</th>
<th>optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>1.0</td>
<td>3.1</td>
<td>3.1</td>
</tr>
<tr>
<td>UV</td>
<td>1.0</td>
<td>2.7</td>
<td>2.7</td>
</tr>
<tr>
<td>DIV</td>
<td>1.0</td>
<td>2.1</td>
<td>2.1</td>
</tr>
<tr>
<td>UV&amp;DIV</td>
<td>1.0</td>
<td>2.4</td>
<td>2.4</td>
</tr>
</tbody>
</table>

**GPU Experiments (Tesla K20c):**

<table>
<thead>
<tr>
<th></th>
<th>no fusion</th>
<th>hand-tuned</th>
<th>optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>1.0</td>
<td>2.3</td>
<td>2.3</td>
</tr>
<tr>
<td>UV</td>
<td>1.0</td>
<td>2.1</td>
<td>2.1</td>
</tr>
<tr>
<td>DIV</td>
<td>1.0</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>UV&amp;DIV</td>
<td>1.0</td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

m = measured time [ms]  
e = estimated time [ms]  
m ~ 1.6e  
m ~ 1.5e

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS’15
Not just your basic, average, everyday, ordinary, run-of-the-mill, ho-hum stencil optimizer

- Complete performance models for:
  - Computation (very simple)
  - Communication (somewhat tricky, using sets and Minkowski sums, parts of the PM)
- Established a stencil algebra
  - Complete enumeration of all program variants
- Navigate the performance space analytically
  - Find the best program variant for a given system
    *Very different for CPU and GPU!*
- Automatic tuning of stencil programs (using the STELLA DS(e)L)
  - 2.0-3.1x speedup against naive implementations
  - 1.0-1.8x speedup against expert tuned implementations

Sponsors:

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS‘15
Backup Slides