TORSTEN HOEFLER

Accelerating weather and climate simulations on heterogeneous architectures

with support of Tobias Gysi, Tobias Grosser, Jeremiah Baer @ SPCL
Scientific **Performance** Engineering

1) Observe

2) Model

3) Understand

4) Build

More at keynote at HPC China tomorrow morning 8:30am!
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

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

\[ a \oplus b = \{ a' + b' | a' \in a, b' \in b \} \]

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

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

Loop Tiling & Loop Fusion
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

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)

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
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
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
Machine Performance Model

- Our model considers peak computation and communication throughputs

![Diagram showing machine performance model with cores, caches, DDR, and communication rates.]

**target machine**
- core 1 (30 Gflop)
  - cache (256 kB)
  - 100 GB/s
  - 25 GB/s
- core 2 (30 Gflop)
  - cache (256 kB)
  - 100 GB/s
  - 25 GB/s
- core 3 (30 Gflop)
  - cache (256 kB)
  - 100 GB/s
  - 25 GB/s

**machine model**
- C = 90 Gflops
- V\(^1\) = 300 GB/s
- L\(^1\) = 50 GB/s
- M\(^1\) = 256 kB
- V\(^0\) = 10 GB/s
- L\(^0\) = 0 GB/s
- M\(^0\) = 8 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 \( l^1_c, ..., l^m_c \)
  \[ t_g = \sum_{c \in g.\text{child}} \max(t_c, v_c/V^m, l^1_c/L^1, ..., l^m_c/L^m) \]
Stencil Performance Model - Affine Sets and Maps

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

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

apply the out origin vector to the transitive closure of all dependencies

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

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

\[ T_{out} = \{(out, (i_0, i_1)) \rightarrow ([i_0/2], [i_1/2])\} \]

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

\[
\begin{align*}
\text{minimize } & t(x) \\
\text{subject to } & m(x) \leq M \\
\end{align*}
\]

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

CPU Experiments (i5-3330):
- no fusion
- hand-tuned
- optimized

GPU Experiments (Tesla K20c):
- no fusion
- hand-tuned
- optimized

m = measured time [ms]
e = estimated time [ms]

T. Gysi, T. Grosser, TH: MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures, ACM ICS'15
From GPUs to the cluster!

CUDA
- over-subscribe hardware
- use spare parallel slack for latency hiding

MPI
- host controlled
- full device synchronization
Latency hiding at the cluster level?

**dCUDA (distributed CUDA)**
- unified programming model for GPU clusters
- avoid unnecessary device synchronization to enable system wide latency hiding

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
dCUDA extends CUDA with MPI-3 RMA and notifications

```c
for (int i = 0; i < steps; ++i) {
    for (int idx = from; idx < to; idx += jstride)
        out[idx] = -4.0 * in[idx] + in[idx + 1] + in[idx - 1] +
                   in[idx + jstride] + in[idx - jstride];

    if (lsend)
        dcuda_put_notify(ctx, wout, rank - 1,
                          len + jstride, jstride, &out[jstride], tag);
    if (rsend)
        dcuda_put_notify(ctx, wout, rank + 1,
                          0, jstride, &out[len], tag);

    dcuda_wait_notifications(ctx, wout,
                             DCUDA_ANY_SOURCE, tag, lsend + rsend);

    swap(in, out); swap(win, wout);
}
```

- iterative stencil kernel
- thread specific idx
- map ranks to blocks
- device-side put/get operations
- notifications for synchronization
- shared and distributed memory

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Hardware supported communication overlap

traditional MPI-CUDA


dCUDA

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Implementation of the dCUDA runtime system

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Overlap of a copy kernel with halo exchange communication

- Benchmarked on Greina (8 Haswell nodes with 1x Tesla K80 per node)

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Weak scaling of MPI-CUDA and dCUDA for a stencil program

- Benchmarked on Greina (8 Haswell nodes with 1x Tesla K80 per node)

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Weak scaling of MPI-CUDA and dCUDA for a particle simulation

- Benchmarked on Greina (8 Haswell nodes with 1x Tesla K80 per node)

![Graph showing execution time vs. number of nodes for MPI-CUDA and dCUDA with halo exchange.](image)

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Weak scaling of MPI-CUDA and dCUDA for sparse-matrix vector multiplication

- Benchmarked on Greina (8 Haswell nodes with 1x Tesla K80 per node)

![Graph showing the execution time of MPI-CUDA and dCUDA with increasing number of nodes.](image)

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
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
- Analytic tuning of stencil programs (using STELLA)
  - 2.0-3.1x speedup against naive implementations
  - 1.0-1.8x speedup against expert tuned implementations
- dCUDA enables overlap of communication and computation
  - Similar to the throughput computing/CUDA idea, just distributed memory
  - Also simplifies programming (no kernel/host code separation)

Sponsors:

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

T. Gysi, J. Baer, TH: dCUDA: Hardware Supported Overlap of Computation and Communication, ACM/IEEE SC16 (preprint at SPCL page)
Backup Slides