Domain-Specific Programming for Climate and Weather

Nabeeh Jumah, Julian Kunkel

Scientific Computing
Department of Informatics
University of Hamburg

SPPEXA Final Symposium 2019
Dresden, Germany
23-10-2019

Project AIMES

Advanced Computation and I/O Methods for Earth-System Simulations

AIMES

- ***Enhance programmability and performance-portability***
- Overcome storage limitations
- Shared benchmark for icosahedral models

Funded within the DFG priority programme SPPEXA

Earth System Modeling

- Models apply numerical methods to simulate earth system
  - Hundreds or thousands of stencils are executed
  - A sequence of stencils is applied each time step

Complexity and Variation Across Models

- Problem domain and grids
  - Dimensions
  - Structure of grids and connectivity
  - Field Localization: staggered vs. collocated grids
- Stencil variability
  - Dimensions
  - Point count
  - Shape
  - Operations

Earth-System Modeling

Performance & Modeling using General-Purpose Languages

- Semantical aspects limit optimization by compilers
- Manual optimization is challenging
  - The complexity of the architectural features
  - The diversity of the architectures
  - Various tools and programming models
- Code quality is harmed: duplication & complexity
- Main considerations arise
  - Code readability and maintainability
  - Developers productivity
  - Performance-portability

Our DSL Approach

- Keep using preferred modeling language
- Extend the modeling language grammar
  - Based on scientific concepts
  - Hiding machine details
- Use semantics of extensions to guide optimization

Separation of Concerns

- Domain scientists formulate scientific problem in source code
- Scientific programmers write target-specific configurations

- Translate code and apply optimization by light-weight tools
  - Extract semantics from source code
  - Use target-specific configuration within separate files
  - Match semantics with config to apply transformations
- Allow users to adapt extensions to model needs

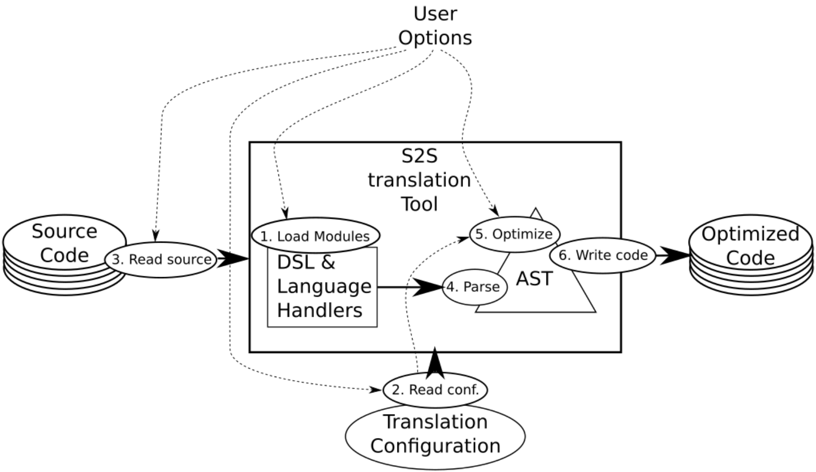
User-Controlled Code Translation

- User-defined language extensions
  - Syntax
  - Behavior
- Maximize semantical impact

Examples

- Example spaecifier group definition:
  SPECIFIER(dim=3D|2D)
  - Defines a dimension specifier group that informs whether the variable represents a 2D or 3D field
- Example access operator definition:
  above(): height=$height+1
  - Allows access to the element directly above the current

Translation process



Refer to: *Performance Portability of Earth System Models with User-Controlled GGDML code Translation (Jumah and Kunkel)*
DOI: 10.1007/978-3-030-02465-9_50

GGDML

GGDML

- **GGDML**:*__G__eneral __G__rid __D__efinition and __M__anipulation __L__anguage*
- Grid definition
- Field declaration
- Field data access/update
  - Iterators
  - Access operators
- Stencil operations

GGDML: Icosahedral Models Language Extensions (Nabeeh Jumah et. al)
DOI: 10.15379/2410-2938.2017.04.01.01

An Example GGDML Code

```
foreach e in grid {

    f_F[e] = f_U[e] * (f_H[e.east_cell()] +
                       f_H[e.west_cell()]) / 2.0;
}

foreach e in grid {

    f_G[e] = f_V[e] * (f_H[e.north_cell()] +
                       f_H[e.south_cell()]) / 2.0;
}
```

Now apply the transformation for a configuration

- OpenMP, MPI/GPU, MPI/OpenMP, ...
- Here: for OpenMP only

Resulting Code for OpenMP

```
for (size_t blk_start = (0); blk_start < (GRIDX + 1);
    blk_start += 20000) {
...
#pragma omp parallel for num_threads(36)
  for (size_t YD_index = (0); YD_index < (local_Y_Eregion);
      YD_index++) {
#pragma omp simd
    for (size_t XD_index = blk_start; XD_index < blk_end;
        XD_index++) {
      f_F[YD_index][XD_index] =
          f_U[YD_index   ][XD_index   ] * (
          f_H[YD_index   ][XD_index   ] +
          f_H[YD_index   ][XD_index -1]) /2.0;
      f_G[YD_index][XD_index] =
          f_V[YD_index   ][XD_index   ] * (
          f_H[YD_index   ][XD_index   ] +
          f_H[YD_index -1][XD_index   ]) /2.0;
    }
  }
}
```

Inter-Kernel Optimization

- Inter-kernel optimization opportunities (e.g., cache reuse)
- Use tools to translate GGDML code and apply optimization
- Allow scientists to control the process

User-Controlled Tool-Supported Procedure

- Automatize the time consuming and complicated parts
  - Tools analyze code
  - Prepare a list of possible fusions
  - Apply fusions selected by scientists
- Maximize possibilities by inter-module optimization
  - Calls are analyzed across code files by tools
  - A list of possible call inlinings is prepared
  - Tools inline calls selected by scientists

Experimental Results for GPU and CPU Code

Architecture	Theoretical Memory bandwidth (GB/s)	Before merge		After merge	
		Measured memory throughput (GB/s)	GFLOPS	Measured memory throughput (GB/s)	GFLOPS
Broadwell	77	62	24	60	31
P100 GPU	500	380	149	389	221
NEC Aurora	1,200	961	322	911	453

Refer to: *Optimizing Memory Bandwidth Efficiency with User-Preferred Kernel Merge (Jumah and Kunkel)*
Test code available at `https://github.com/aimes-project/ShallowWaterEquations`

Multi-Node Parallelization

Data Access

- How is the problem domain decomposed
- Which operations need which data
- Where to find that data
- How to make data available for computation

Explicit Memory Data Access

- Developers take care
- Application code includes necessary details
  - Map global points to local (subdomain mapping)
  - Which data on which node
  - Indices to access local memory on each node

Our Approach – MODA

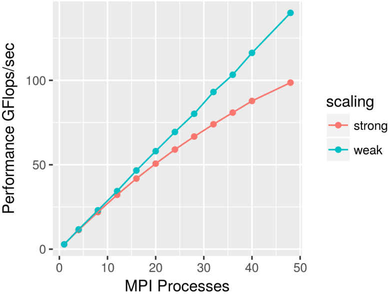
- Source code with scientific concepts
- Code unaware of hardware
  - Single vs. multiple nodes
  - Memory; shared vs. distributed, host vs. device ...
  - Processors; multi-core vs. GPU v.s VE vs. ...

Memory-Oblivious Data Access (MODA)

- Get rid of explicit tracking of data location
  - No node location
  - No array indices
- Alternative indices
  - Scientific basis; e.g. spatial relationships
  - Unaware of underlying memory and hardware

Experimental Results

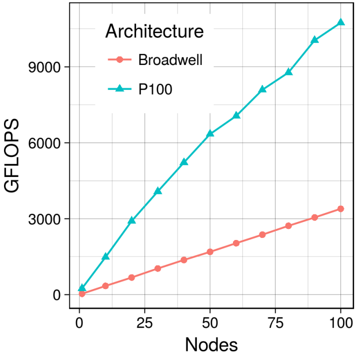
Figure: Scalabilty experiments (Triangular unstructured grid)



Refer to: *Performance Portability of Earth System Models with User-Controlled GGDML code Translation (Jumah and Kunkel)*
DOI: 10.1007/978-3-030-02465-9_50

Experimental Results

Figure: Scalability experiments (Structured grid)
Shallow water equation solver



Refer to: *Scalable Parallelization of Stencils using MODA (Jumah and Kunkel)*
Test code available at `https://github.com/aimes-project/ShallowWaterEquations`

Memory Layout, Loop Nests, & Vectorization

- MODA hides actual data location in memory
- Our techniques allow flexible layout transformations
  - Simple index intechange
  - Or whatever formula to define data location
- Loop order control allows optimal access besides data layout
- Vectorization needs a corresponding data layout & loop order

Memory Layout, Loop Nests, & Vectorization

Table: Data layout experiments (Triangular unstructured grid)

	Performance (GFLOPS)		
	Serial	P100	V100
3D	1.97	220.38	854.86
3D-1D	1.99	408.15	1240.19

Refer to: *Performance Portability of Earth System Models with User-Controlled GGDML code Translation (Jumah and Kunkel)*
DOI: 10.1007/978-3-030-02465-9_50

Table: Array-stride experiments (Structured grid)

Architecture	GFLOPS		
	Scattered	Short distance	Contiguous
Broadwell	3	13	25
NEC Aurora	80	161	322

Refer to: *Automatic Vectorization of Stencil Codes with the GGDML Language Extensions (Jumah and Kunkel)*
DOI: http://doi.acm.org/10.1145/3303117.3306160
Test code available at `https://github.com/aimes-project/ShallowWaterEquations`

Conclusion

- GGDML provides semantics to drive optimization
- GGDML simplifies model development
  - Scientists write scientific code
  - Optimization is driven by separate configuration files
- Using GGDML we could apply diffent optimization techniques
  - Kernel optimizations
  - Inter-kernel optimizations
  - Multi-node parallelization
- Using GGDML exactly one code version is written
- GGDML code is performance portable

Acknowledgement

- DFG (German Research Foundation)
- German Climate Computing Center (DKRZ)
- Swiss National Supercomputing Center (CSCS)
- Erlangen regional computing center (RRZE) at Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
- NEC Deutschland
- Prof. John Thuburn, University of Exeter