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

Julian M. Kunkel, Thomas Ludwig, Thomas Dubos, Naoya Maruyama, Takayuki Aoki, Günther Zängl, Hisashi Yashiro, Ryuji Yoshida, Hirofumi Tomita, Masaki Satoh, Yann Meurdesoif, Nabeeh Jum'ah, Anastasiia Novikova, Anja Gerbes (**Contact:** juliankunkel@gmail.com)

Motivation

- Several groups work on icosahedral-grid based climate/weather models
- Obstacles for Exascale simulations - but also on small scale:
  - Code is very complex and difficult to refactor
  - Climate prediction creates huge data volumes

Limitations of general-purpose programming languages

- Semantics and syntax restrict programmers productivity
- Performance is hardly portable between architectures

Existing Domain-Specific Languages

- May create optimized code for different architectures
- Technical languages with limited relation to scientific domain
- Typically require language-specific paradigm shift for scientists
- Unclear future of the framework/tool

Existing scientific file formats

- Metadata for icosahedral data is not standardized
- Difficult to achieve good performance
- Pre-defined compression schemes achieve suboptimal ratio

Goals

Address issues of icosahedral earth-system models

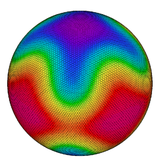
- Enhance programmability and performance-portability
- Overcome storage limitations
- Provide a common benchmark for icosahedral models

Collaboration

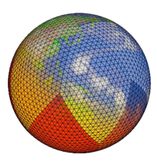
Funded partners

Thomas Ludwig (Universität Hamburg)
Thomas Dubos (Institut Pierre Simon Laplace)
Naoya Maruyama (RIKEN)
Takayuki Aoki (Tokio Institute of Technology)

ICON



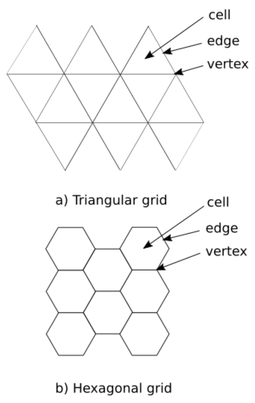
DYNAMICO



NICAM

GGDML Domain-Specific Language

- **GGDML**: the ***G****eneral* ***G****rid* ***D****efinition and* ***M****anipulation* ***L****anguage*
- Abstracted scientific-domain based constructs for:
  - Data types reflecting "grid" concepts
  - Variable declaration & allocation
  - Iterators to traverse and update variables
  - Named neighbours in (triangular/hexagonal) grids
- Developed in co-design with domain scientists



Fortran code (dynamico) and GGDML version

```
DO l=ll_begin,ll_end
!DIR$ SIMD
  DO ij=ij_begin,ij_end
   berni(ij,l) = .5*(geopot(ij,l)+geopot(ij,l+1)) + 1/(4*Ai(ij)) *
       (le(ij+u_right)*de(ij+u_right)*u(ij+u_right,l)**2 &
       +le(ij+u_rup)  *de(ij+u_rup)  *u(ij+u_rup,l)**2   &
       +le(ij+u_lup)  *de(ij+u_lup)  *u(ij+u_lup,l)**2   &
       +le(ij+u_left) *de(ij+u_left) *u(ij+u_left,l)**2  &
       +le(ij+u_ldown)*de(ij+u_ldown)*u(ij+u_ldown,l)**2 &
       +le(ij+u_rdown)*de(ij+u_rdown)*u(ij+u_rdown,l)**2 )
  ENDDO
ENDDO
```

GGDML version of the code above

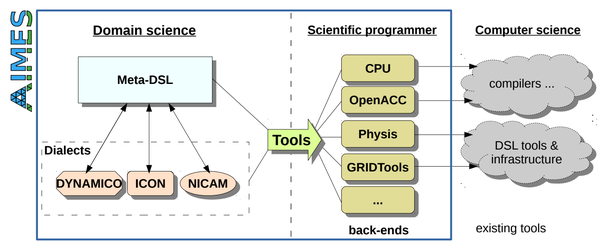
```
FOREACH cell IN grid
 berni(cell) = .5*(geopot(cell)+geopot(cell%above)) + 1/(4*Ai(cell%ij)) *
  REDUCE(+, N={1..6}
    le(cell%neighbour(N)%ij)*de(cell%neighbour(N)%ij)*u(cell%neighbour(N))**2)
END FOREACH
```

- Higher-level code is translated into target-architecture-optimized code, driven by
  - The semantics of the GGDML extensions
  - User-provided architecture-specific configurations

Scientific Work Packages: Objectives and Tasks

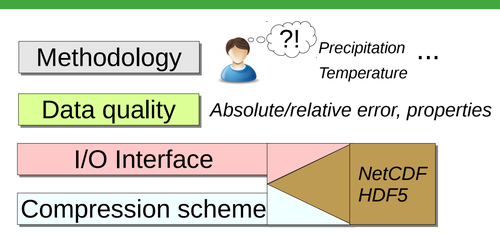
WP 1: Towards higher-level code design

- Foster separation of concerns: Domain scientists, scientific programmer and computer scientists
  - High level of abstraction, reflects domain science concepts
  - Independence of hardware-specific features, e.g. memory-layout
  - Convertible into existing languages and DSLs
- 1.1-1.3 Develop/reformulate key parts of models into DSL-dialects
- 1.4 Design common DSL concepts for icosahedral models
- 1.5 Develop source-to-source translation tool and mappings



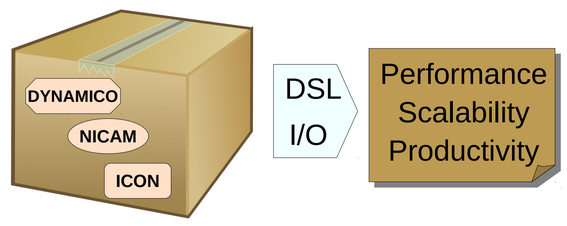
WP 2: Massive I/O

- 2.1 Optimize file formats for icosahedral data
- 2.2 Data reduction concepts
- 2.3 API for user-defined variable accuracy
- 2.4 Identifying required variable accuracy
- 2.5 Lossy compression



WP 3: Evaluation

- 3.1 Selection of representative test cases
- 3.2 Extraction of simple kernels
- 3.3 Common benchmark package/mini-IGCMs[1]
- 3.4 Benefit of the DSL for kernels/mini-IGCMs
- 3.5 Estimating benefit for full-featured models
- 3.6 I/O advances for full models

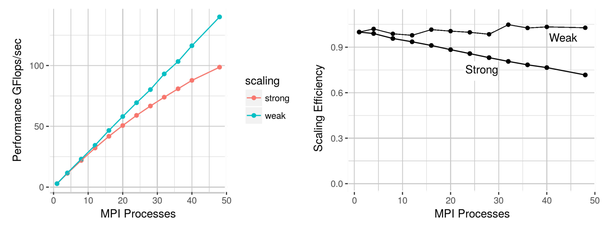


Project's Timeframe and Status

- Started March 2016 and ends February 2019
- WP 1 status:
  - Dialect development: delivered May 2017.
  - The development of the DSL: delivering March 2018.
  - The source-to-source translation tool: in progress.

Higher-Level Code

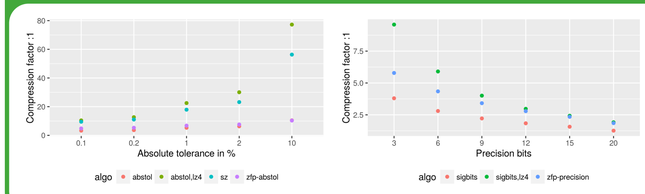
- GGDML code is currently translated into different targets
  - Multicore processors (with OpenMP)
  - Multi-node with Multicore processors (OpenMP+MPI)
  - GPU-accelerated machines (with OpenACC)
  - Multi-node with GPUs (OpenACC+MPI)



- Automatic domain decomposition for MPI

- WP 2 status:
  - Extended compression library with new algorithms : delivered 2017
  - Definition of all quantities : delivered 2017
  - Integration into HDF5/NetCDF4 : delivered Jan. 2018

Tolerance-Based Results



- WP 3 status:
  - IcoAtmosBenchmark v.1 (kernel suites): delivering March 2018.
  - IcoAtmosBenchmark v.2 (mini-apps): in progress.

Compiler Optimization

- Aim is to optimize compilers that HPC applications can run more efficiently
- Provide an interface to the DSL to make transformations that a compiler may not make due to language restrictions
- Optimize HPC patterns that can achieve the best possible performance from the compiler

Compression

- Development of Scientific Compression Library
  https://github.com/JulianKunkel/scil
- Users define the required accuracy
  - In terms of relative/absolute/precision ...
  - In terms of required performance
  - The library picks a fitting algorithm
- Fill value integration into existing algorithms
- Extending the library with new compression algorithms
- Integration into HDF5 / NetCDF4
- Testing with different climate models: Isabel, ECHAM, NICAM

Extending NICAM with a high-level framework

- GridTools
  - C++ template framework for weather and climate models
  - Architecture-independent programming interface for performance and portability
- Evaluating GridTools as a programming framework for NICAM
- Successfully ported representative NICAM stencil kernels with comparable performance as hand-tuned implementations

Further Planned Optimization

- Inter-kernel optimization
  - Experiments to evaluate optimization impact
  - Start with manual optimization experiments
  - Improve the source-to-source translation tool to apply such optimization within the code translation process
- Loop blocking improvement

Acknowledgement

This work was supported by the German Research Foundation (DFG) through the Priority Programme 1648 „Software for Exascale Computing" (SPPEXA).

https://wr.informatik.uni-hamburg.de/research/projects/aimes/start