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

Julian M. Kunkel, Nabeeh Jumah, Anastasiia Novikova, Thomas Ludwig, Thomas Dubos, Sunmin Park, Hisashi Yashiro, Günther Zängl, John Thuburn

Scientific Computing
Department of Informatics
University of Hamburg

2019-01-23

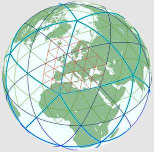
AIMES

Goals

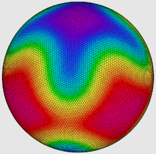
Address key issues of icosahedral earth-system models

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

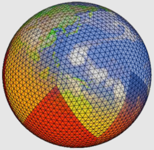
Covered models



ICON



DYNAMICO



NICAM

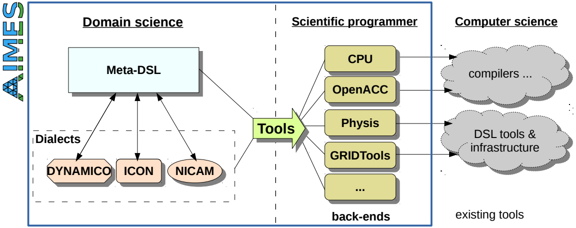
WP1: Towards Higher-Level Code Design

Recap: Goals of the WP

- Bypass shortcomings of general-purpose languages
  - Offer performance-portability
  - Enhance source repositories maintainability
  - Get rid of complexity in optimized-code development
  - Enhance code readability and scientists productivity
- Extend modelling programming language
  - Based on domain science concepts
  - Free of lower level details (e.g., architecture, memory layout)

Approach

- Foster separation of concern
  - Domain scientists develop domain logic in source code
  - Scientific programmers write hardware configurations
- Source code written with extended language
  - Closer to domain scientists logic
  - Scientists do not need to learn optimizations
  - Write code once, get performance for various configurations
- Hardware configurations define software performance
  - Written by programmers with more experience in platform
  - Comprise information on target run environment



Progress

- Achievements of the first year
  - Evaluation of GridTools for NICAM
  - HybridFortran support in ASUCA
  - Development of the model dialects and extensions
  - Implementing a basic source-to-source translation tool
  - Evaluating the DSL's impact on programmability
    Refer to: Nabeeh Jumah et al. "GGDML: Icosahedral Models Language Extensions". In: Journal of Computer Science Technology Updates. Volume 4, Number 1 (June 2017)
- Achievements of the second year
  - Refinements to the language extensions
  - Implementing more features in the translation tool
    - Configurable language extensions
    - Configurable memory layout
    - Configurable parallelization
    - Configurable halo exchange
  - Experiments on the performance and performance portability

Progress of the Third Year

- Achievements of the third year
  - Developing new test application as proxy application
    - Shallow water equations
    - Structured grid
  - Implementing more features in the translation tool to aid dev.
    - User-configuration of selected optimizations
      e.g. kernel merging to enable memory re-use
    - Configurable loop interchange and blocking
    - Supporting user-guided domain decomposition
    - Annotating kernels for Likwid instrumentation
    - Automatic generation of code to handle halo dirty regions
  - More experiments on performance (scaling, optimizations)
  - Exploring another architecture (NEC Aurora vector engine)
  - Developing a prototype for GASPI (to be evaluated)

GGDML Code Example

```
foreach c in grid
{
  float df=(f_F[c.east_edge()]-f_F[c.west_edge()])/dx;
  float dg=(f_G[c.north_edge()]-f_G[c.south_edge()])/dy;
  f_HT[c]=df+dg;
}
```

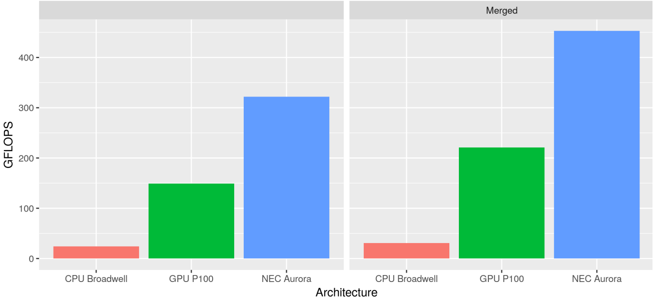
A sample generated C code for OpenMP + MPI

```
    ...handle domain decomposition and halo mangagement
  for (size_t blk_start = (0); ... blocking
    size_t blk_end = ...
#pragma omp parallel for
    for (size_t YD_index = 0; YD_index < local_Y_Cregion;
        YD_index++) {
#pragma omp simd
      for (size_t XD_index = blk_start; XD_index < blk_end;
          XD_index++) {
        float df = (f_F[YD_index][XD_index +1] -
                    f_F[YD_index][XD_index]) /dx;
        float dg = (f_G[YD_index +1][XD_index] -
                    f_G[YD_index][XD_index]) /dy;
        f_HT[YD_index][XD_index] = df + dg;
      }
```

Performance Evaluation

- Test application
  - Shallow water equations
  - Structured grid
- Test machines
  - Intel(R) Xeon(R) CPU E5-2695 v4 with 2.10GHz
    - Level 3 cache is 45 MB (shared among 18 cores)
  - Tesla P100 GPUs
  - NEC SX-Aurora TSUBASA vector engine
- Experiments
  - Efficiency on different architectures
  - Performance depending on kernel blocking

The Effect of Kernel Merging on Performance

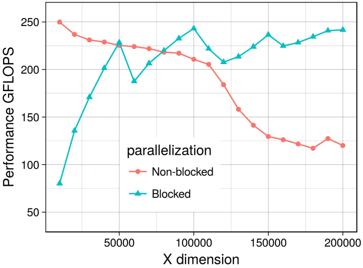


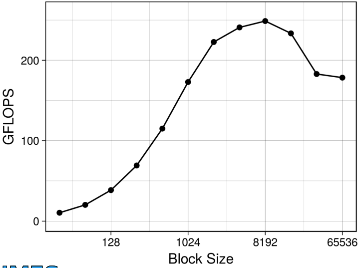
Architecture	Theoretical Memory bandwidth (GB/s)	Before merge		After merge	
		Measured memory throughput (GB/s)	GFLOPS	Measured memory throughput (GB/s)	GFLOPS
Broadwell	77	62	24	60	31
P100 GPU	500	380	149	389	221
NEC Aurora	1,200	961	322	911	453

All cases utilize between 76 and 80% memory bandwidth

Evaluation - Blocking on P100 GPUs

- Performance drop is steep around grid width of 130K as a result of the cache size
- Blocking stabilizes performance over wider grids (grid width >80K)
- Blocking harms performance for smaller grids
- The DSL can emit both





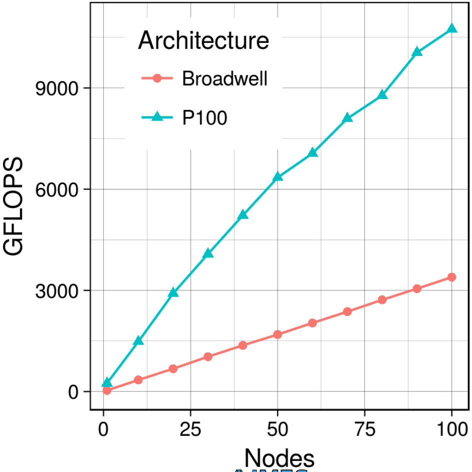
Evaluation - Vectorization and Memory Layout

- Investigation of vectorization and memory layout alternatives
- Tested three cases
  - Scattered elements: distant elements
  - Constant short distance: 4 bytes between consecutive elements
  - Contiguous (unit-stride) array
- We show impact of matching memory layout and access on
  - Vector unit and instructions utilization
  - Memory bandwidth utilization efficiency
- Allowed to simulate AoS performance

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

Evaluation - Scalability

- Investigation of scalability on 1,10,20..100 nodes
- Tested on Broadwell and P100 GPUs



Outlook Until the End of AIMES

- Investigate using GASPI as alternative for communication
  - Performance and other considerations
  - Differences to MPI
- Investigate semi-structured grids (again from Y1)
  - Prepare necessary configurations
  - Investigate halo exchange optimizations
  - Investigate vectorization considerations
  - Compare to structured grids
- Make the tool available on GitHub
- Provide tool for brute-force optimization exploration

Massive I/O

Recap: Goals of the WP2

- Optimization of I/O middleware for icosahedral data
  - Throughput, metadata handling
- Design of domain-specific compression (ratio $> 10 : 1$)
  - Investigate metrics allowing to define accuracy per variable
  - Design user-interfaces for specifying accuracy
  - Develop a methodology for identifying the required accuracy
  - Implement compression schemes exploiting this knowledge

Progress

- Achievements of the first year
  - C-API Design of scientific compression interface library (SCIL)
    - Quantities
    - Tools
    - Compression chain
  - Evaluation on synthetic data
  - Evaluation on scientific data (cloud model ECHAM)
- Progress in the second year
  - Survey of file formats
  - Refactoring (Project structure, quantities)
  - New tool: Pattern creator
  - HDF5 compression plug-in
  - Evaluation on synthetic data patterns
  - Evaluation on scientific data (hurricane model Isabel)

Recent Progress in Year Three

- Evaluating compression ratios/performance with NICAM files
- Integration of SCIL into NICAM
- Improved compatibility on various layers and tools
- Alternative specification of compression characteristics (in an external file)
- Investigated incremental compression algorithm
- Explored science case to identify tolerable precision loss

Configuration File for the Quantities

The file in the env. var. NETCDF_SCIL_HINTS_FILE is read by NetCDF and applied to the respective variables:

```
Variable1:
 relative_tolerance_percent=1

Variable2:        # Developer comment1
 relative_tolerance_percent=1
 relative_err_finest_abs_tolerance=1
 absolute_tolerance=1
 significant_digits=1
 significant_bits=1
 lossless_data_range_up_to=1
 lossless_data_range_from=1
 fill_value=4711
 comp_speed=0.5*MiB
 decomp_speed=1*NetworkSpeed
 force_compression_methods=abstol,lz4
```

Investigate Tolerable Error

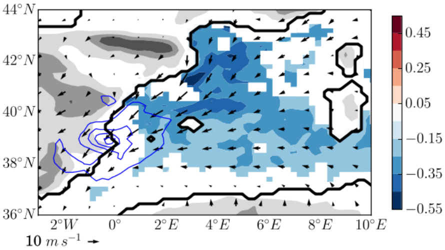
- Under noise mimicking compression, reproduce conclusions of Berthou et al. 2016[1]
  - Mediterranean region (Spain, France)
  - Evaluates how wind, through its action on the ocean, impacts (with some delay) heavy precipitation
  - Compares outputs from two simulations (CPL and SMO) and subjects the difference to tests of statistical significance
    Student t test; 97.5% probability of rejecting a zero difference
  - Analyzed fields: wind, rain (convective and non-convective), humidity, sea-surface temperature (SST)

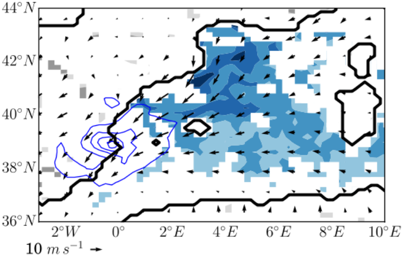
[1] DOI 10.1007/s00382-016-3153-0

Approach

1. Reproduce published results (Find data/scripts used in paper)
2. Apply statistical model for noise induced by lossy compression
   - Use Gaussian white noise
3. Redo the analysis, check if the conclusions are still supported
4. Increase levels of noise to input data (= model output)
   - One field at a time; then together

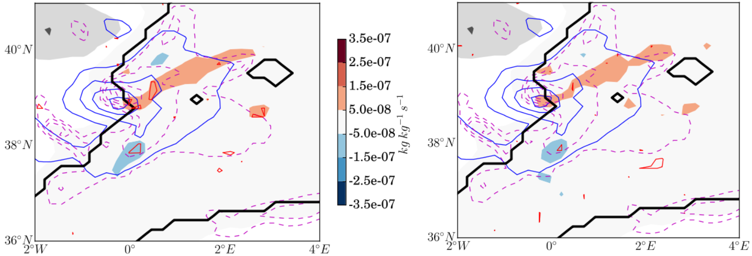
Compression Error Propagation: SST





- Blue shade: conditional mean SST difference between CPL and SMO simulations where statistical test is passed
- Adding noise affects little the overall pattern, but makes it harder to pass statistical test (more white holes in the blue)
- Conclusions unchanged even with quite large noise on SST (0.2°K), probably due to averaging several events

Compression Error Propagation: Wind



- Blue/orange shade = conditional mean difference between CPL and SMO of moisture flux divergence
  Statistical test is passed inside red contours
- Noise makes it harder to pass statistical test
- Despite noise added to wind, moisture flux divergence still OK
- Noise impact expected to worsen at higher resolution
- Projects like CORDEX2 advocate for compression methods that control noise spectrum (which is part of AIMES)

Outlook Until the End of AIMES

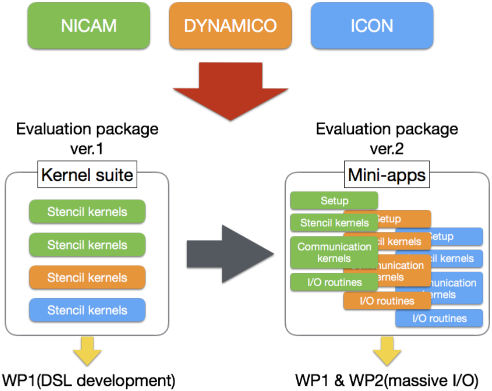
- Full application I/O benchmark with NICAM (& SCIL)
- Pushing HDF5-plugin for SCIL to HDF5 group
- Wrap-up of the documentation

We'll try to further work with students

- Integration of alternative lossy compression algorithms
- Machine learning of
  - Performance, expected ratio for data
  - Best compression algorithms

WP 3: Evaluation

- Providing benchmark packages from icosahedral weather/climate models
- Evaluating the DSL and domain-specific I/O advancements



Achievements in Year Three

- ICON kernels
  - Selection, Extraction
- ICON-like mini application
  - Written with GGDML
  - C is the host language
- Implemented NetCDF I/O support within this mini app
  - GGDML kernels support I/O
- ICON-like application integration within benchmark

Outlook Until the End of AIMES

- Integration tests
  - DSL
  - I/O
  - Compression
- Update the already published benchmark suite
  - Testcode for GGDML
  - I/O benchmarking results

Summary

- AIMES covers programmability issues on the high-level
  - DSL-extensions enrich existing languages
  - Fosters separation of concerns, improves performance portability
  - A testbed application with different kernels has been developed
  - The translation tool now supports own optimizations
  - Multi-core, GPUs, vector supported for the testcodes
- AIMES addresses domain-specific lossy compression
  - (Help) scientists to define the variable accuracy
  - Exploit this knowledge in the compression scheme
  - Novel schemes compete with existing algorithms

Backup

Backup

Differences among three icosahedral atmospheric models

- Horizontal grid system
  - NICAM: co-located, semi-structured
  - DYNAMICO: staggered, semi-structured
  - ICON: staggered, unstructured

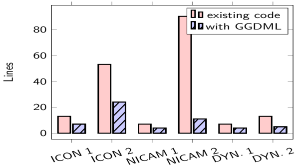
  - semi-structured means... "structured for stencil operation, unstructured for communication topology"

GGDML Impact on the Source Code

The DSL reduces development and maintenance effort

- LOC statistics

Model, kernel	lines (LOC) before DSL	lines (LOC) with DSL	words before DSL	words with DSL	characters before DSL	characters with DSL
ICON 1	13	7	238	174	317	258
ICON 2	53	24	163	83	2002	916
NICAM 1	7	4	40	27	76	86
NICAM 2	90	11	344	53	1487	363
DYNAMICO 1	7	4	96	73	137	150
DYNAMICO 2	13	5	30	20	402	218
total	183	55	911	430	4421	1991
relative size with dsl	30%		47%		45%	



- Predicting saving applying the DSL to 300k code of ICON
  - 100k infrastructure (does not change with the DSL)
  - Remaining code reduced according to our test kernels
  - COCOMO estimations

Software project	Version	Effort Applied	Dev. Time (months)	People require	dev. costs (M€)
Semi-detached		2462	38.5	64	12.3
	DSL	1133	29.3	39	5.7
Organic		1295	38.1	34	6.5
	DSL	625	28.9	22	3.1

Partners and Expertise

Funded partners

- Thomas Ludwig (Universität Hamburg)
  I/O middleware, compression, ICON DSL
- Thomas Dubos (Institut Pierre Simon Laplace)
  Application I/O servers, compression, DYNAMICO
- Naoya Maruyama (RIKEN)
  DSL (Physis), GPUs, NICAM
- Takayuki Aoki (Tokio Institute of Technology)
  DSL (HybridFortran), language extension, peta-scale apps

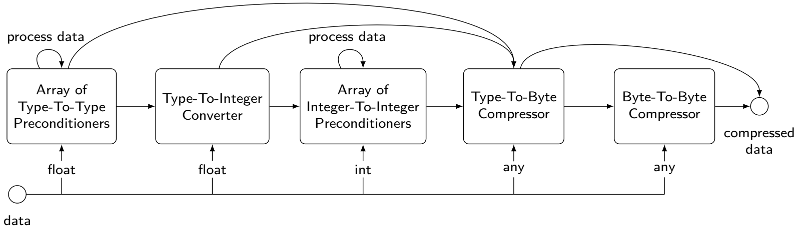
Cooperation Partners

- DKRZ *(I/O, DSL)*
- DWD *(ICON, DSL, I/O)*
- University of Exeter *(Math. aspects in the DSL)*
- CSCS *(GPU/ICON, GRIDTool, compression)*
- Intel *(DSL-backend optimization for XeonPhi, CPU)*
- NVIDIA *(DSL-backend optimization for GPU)*
- The HDF Group *(I/O, unstructured data, compression)*
- NCAR (MPAS developers, another icosahedral model)
- Bull
- Cray

Information exchange, participate in workshops, [hardware access]

Appendix. WP2: Architecture of SCIL

- Contains tools to
  - Create random patterns, compress/decompress, add noise, plot
- HDF5 and NetCDF4 integration; tools support NetCDF3, CSV
- Library with
  - Automatic algorithm selection (under development)
  - Flexible compression chain:



WP2: Supported Quantities

Accuracy quantities:

absolute tolerance: compressed can become true value ± absolute tolerance

relative tolerance: percentage the compressed value can deviate from true value

relative error finest tolerance: value definining the absolute tolerable error for relative compression for values around 0

significant digits: number of significant decimal digits

significant bits: number of significant decimals in bits

Performance quantities:

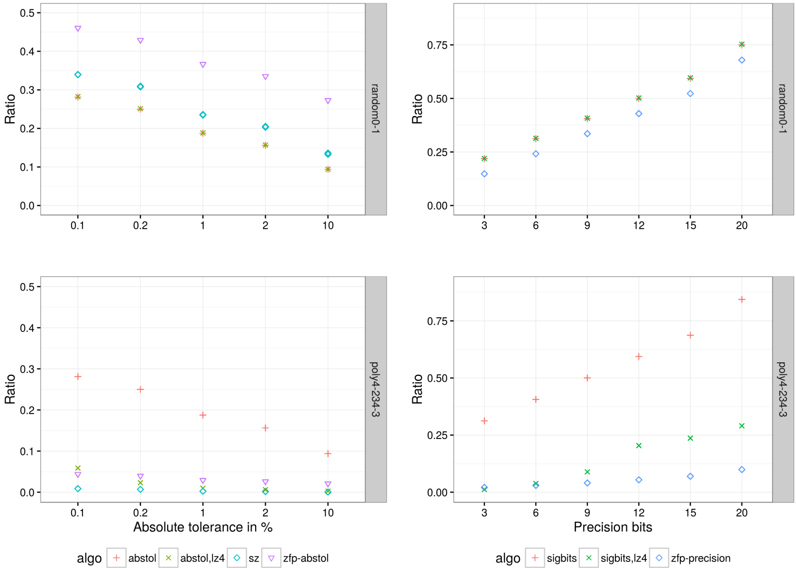
compression speed: in MiB or GiB, or relative to network or storage speed

decompression speed: in MiB or GiB, or relative to network or storage speed

Supplementary quantities:

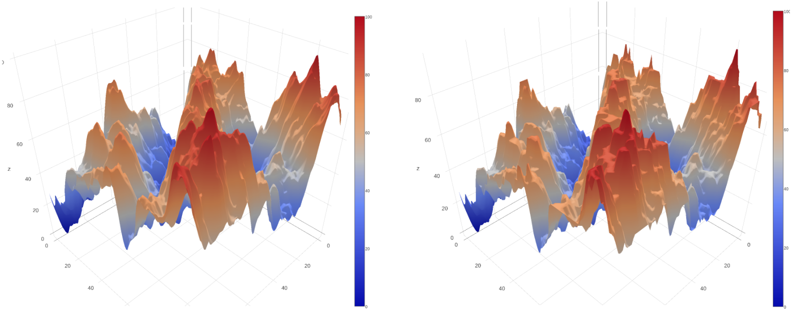
fill value: a value that scientists use to mark special data point

WP2: Synthetic Patterns



WP2: Example Synthetic Data

Simplex (options 206, 2D: 100x100 points)



Right picture compressed with Sigbits 3bits (ratio 11.3:1)

WP2: Analyzing Performance of Lossy Compression

Data

- Single precision (1+8+23 bits)
- Synthetic, generated by SCIL's pattern lib.
  - e.g., Random, Steps, Sinus, Simplex
- Data of the variables created by ECHAM (123 vars), Isabel

Experiments

- Single thread, 10 repeats
- Lossless (memcopy and lz4)
- Lossy compression with significant bits (zfp, sigbits, sigbits+lz4)
- Lossy compression with absolute tolerance (zfp, sz, abstol, abstol+lz4)
  - Tolerance: 10%, 2%, 1%, 0.2%, 0.1% of the data maximum value

WP2: Comparing Algorithms for the Scientific Files

	Algorithm	Ratio	Compr. MiB/s	Decomp. MiB/s
ECHAM	abstol	0.190	260	456
	abstol,lz4	0.062	196	400
	sz	0.078	81	169
	zfp-abstol	0.239	185	301
Isabel	abstol	0.190	352	403
	abstol,lz4	0.029	279	356
	sz	0.016	70	187
	zfp-abstol	0.039	239	428
Random	abstol	0.190	365	382
	abstol,lz4	0.194	356	382
	sz	0.242	54	125
	zfp-abstol	0.355	145	241

(a) 1% absolute tolerance

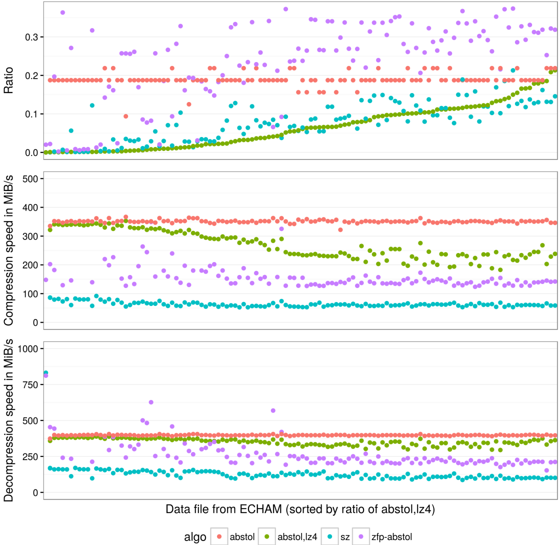
	Algorithm	Ratio	Compr. MiB/s	Decomp. MiB/s
ECHAM	sigbits	0.448	462	615
	sigbits,lz4	0.228	227	479
	zfp-precision	0.299	155	252
Isabel	sigbits	0.467	301	506
	sigbits,lz4	0.329	197	366
	zfp-precision	0.202	133	281
Random	sigbits	0.346	358	511
	sigbits,lz4	0.348	346	459
	zfp-precision	0.252	151	251

(b) 9 bits precision

Table: Harmonic mean compression of scientific data

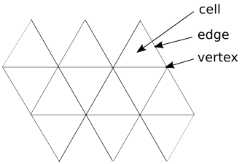
WP2: Results for Absolute Tolerance of ECHAM

Comparing algorithms using an absolute tolerance of 1% of the maximum value

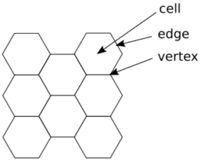


DSL Development

- Co-design with scientists to develop DSL constructs
  - Current version represents several iterations
  - GGDML: *General grid definition and manipulation language*
  - Grid definition
  - Grid-bound variable declaration
  - Grid-bound variable access/update
  - Stencil operations
- Hides memory locations and access details, data iteration
- Abstract higher concepts of grids, hiding connectivity details



a) Triangular grid



b) Hexagonal grid

Progress of WP3

- Achievements in the first year
  - NICAM kernels
    - Selection, Extraction
    - Performance check (on the K computer(RIKEN), Mistral(DKRZ))
  - DYNAMICO and ICON kernels
    - Selection
  - tools for damping the reference data
- Progress of the second year
  - Packaging IcoAtmosBenchmark v.1
  - NICAM kernels
    - Documents
    - Implementation by GridTools
  - DYNAMICO kernels
    - Extraction, Performance check, Documents