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

Nabeeh Jum'ah, Anastasiia Novikova, Julian M. Kunkel, Thomas Ludwig, Thomas Dubos, Naoya Maruyama, Takayuki Aoki, Günther Zängl, Hisashi Yashiro, Ryuji Yoshida, Hirofumi Tomita, Masaki Satoh, Yann Meurdesoif

Scientific Computing
Department of Informatics
University of Hamburg

2018-03-21

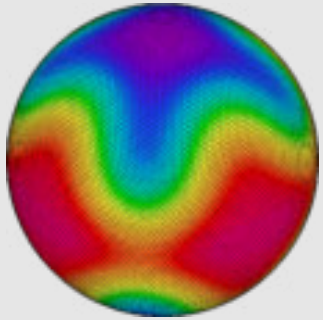
Goals

Address key issues of icosahedral earth-system models

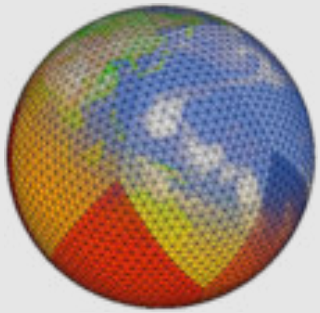
- Enhance programmability and performance-portability
- Overcome storage limitations
- Shared benchmark for these models

Covered models

ICON



DYNAMICO



NICAM

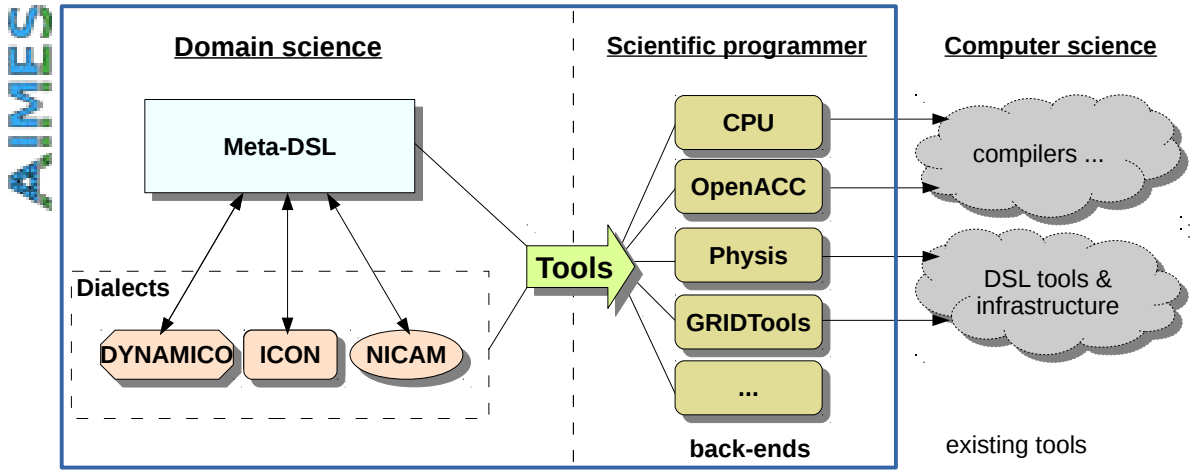
WP1: Towards Higher-Level Code Design

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

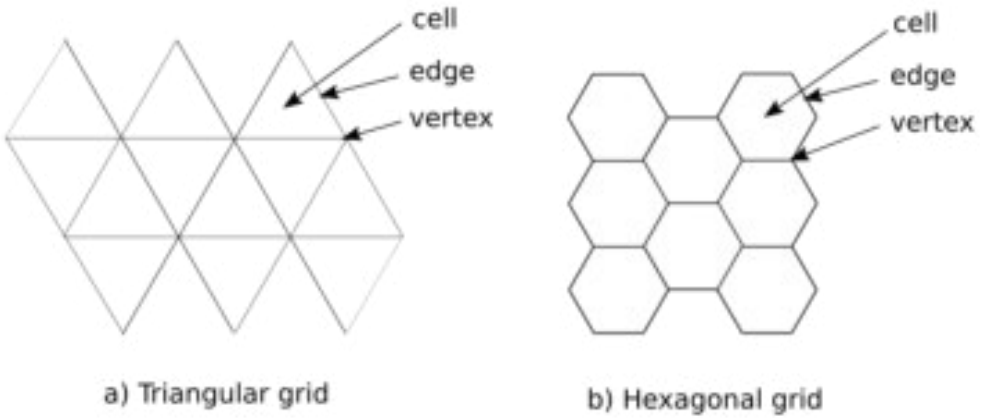


Progress

- Previous achievements (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)
- Recent progress (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

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



Fortran vs. GGDML Code Example

```
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)) * REDUCE(+,N, le(cell%neighbour(N))*
     de(cell%neighbour(N))* u(cell%neighbour(N))**2)
END FOREACH
```

Source-to-Source Translation

- Our translation tool transforms code based on
  - Higher-level semantics of the language extensions
  - Configuration information to control code optimization

Translation Configurations

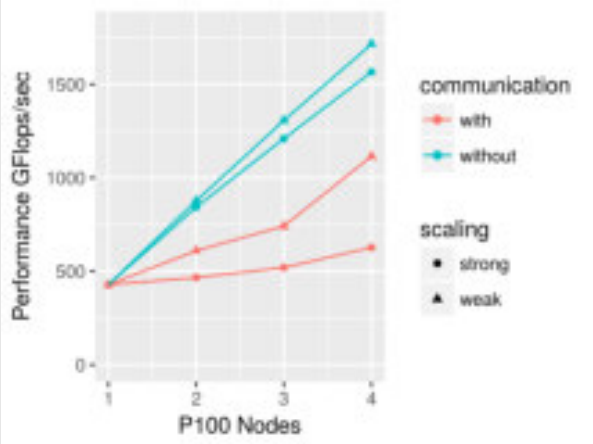
- Define language extensions
  - access specifiers, *e.g., float CELL 3D cell*
  - access operators, *e.g., cell.above*
- Control memory allocation/deallocation
- Define grids
- Control code parallelization
- Control memory layout
- Control halo exchange in multi-node configurations

Performance Evaluation

- Current tool's implementation can transform code into
  - GPU code with OpenACC
  - MPI code on multi-node configurations (MPI+OpenACC)
- The table below shows impact of changing memory layout
  - On P100 and V100 GPUs
  - With 3D array, and a transformed 1D array

Testcode performance on P100 and V100 GPUs

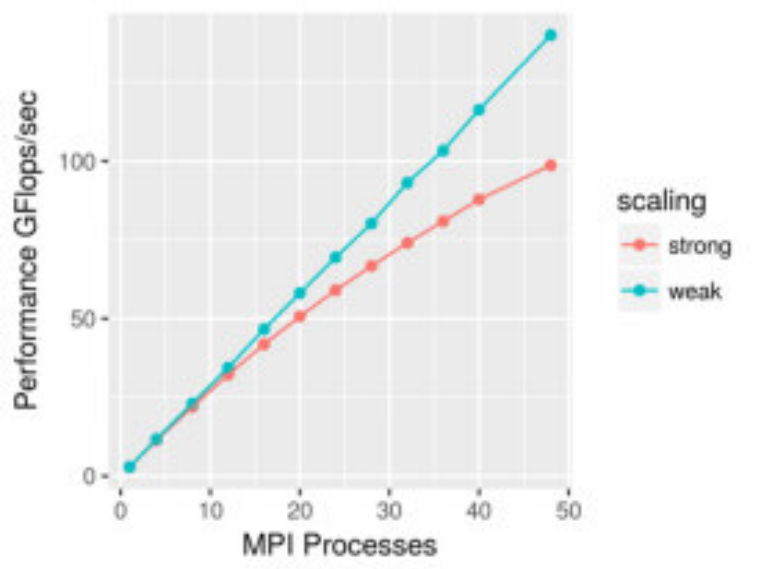
	Serial	P100			V100		
		performance GF/s	Memory throughput GB/s		performance GF/s	Memory throughput GB/s	
			read	write		read	write
3D	1.97	220.38	91.34	56.10	854.86	242.59	86.98
3D-1D	1.99	408.15	38.75	43.87	1240.19	148.49	57.12

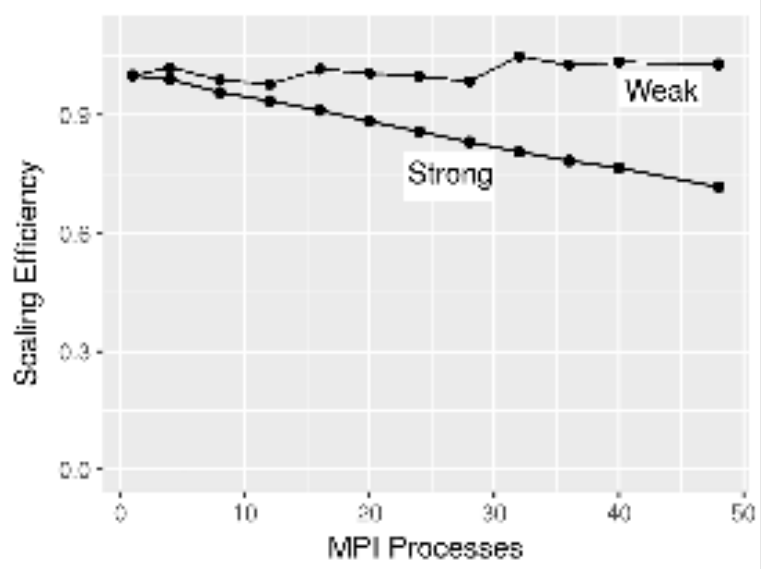


Performance Evaluation

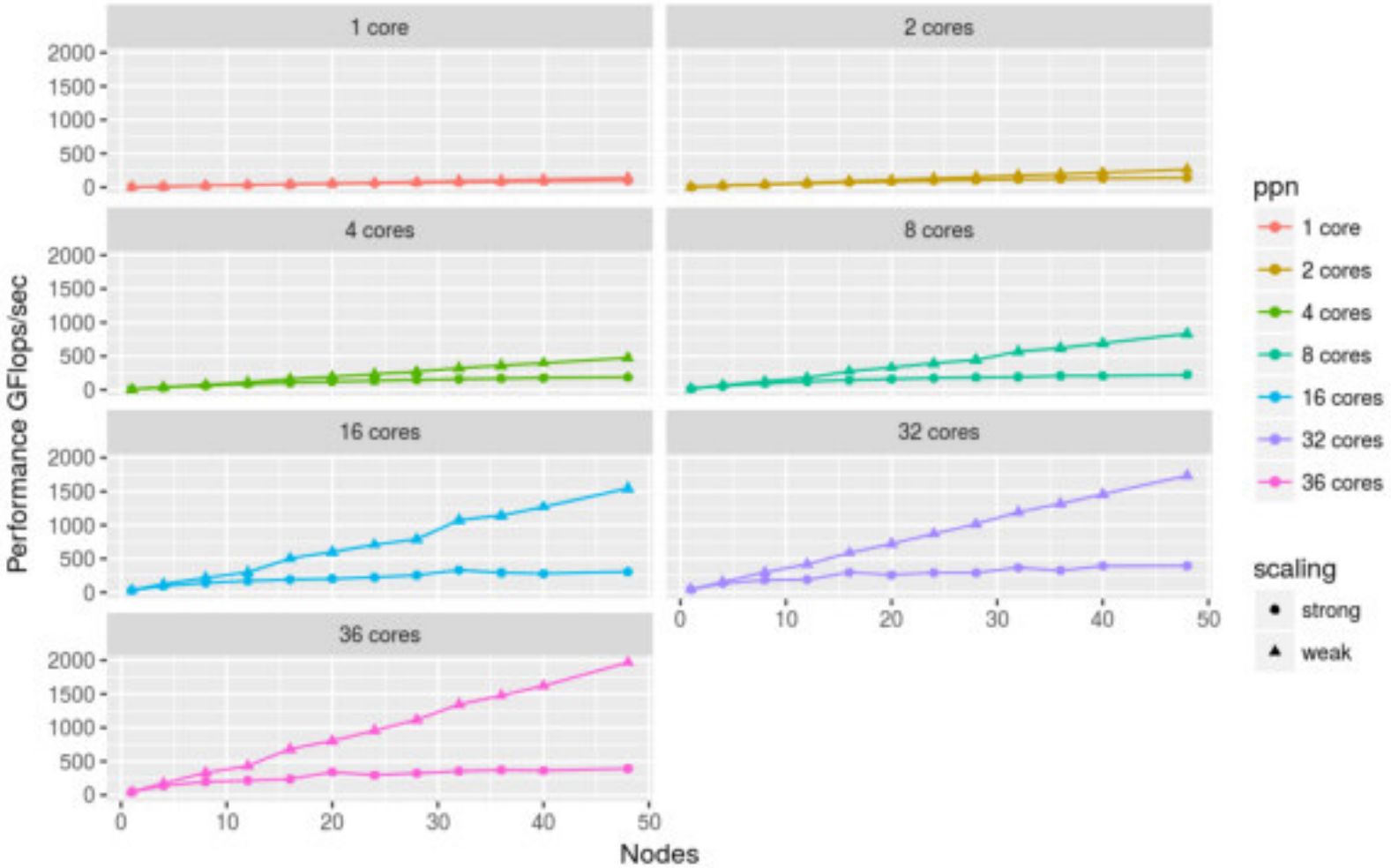
- Results for:
  - Multi-core processor code with OpenMP
  - MPI code on multi-node configurations (MPI+OpenMP)

Testcode performance on Broadwell processors





Performance Evaluation



The testcode scalability under different numbers of MPI processes running different numbers of cores.

Massive I/O

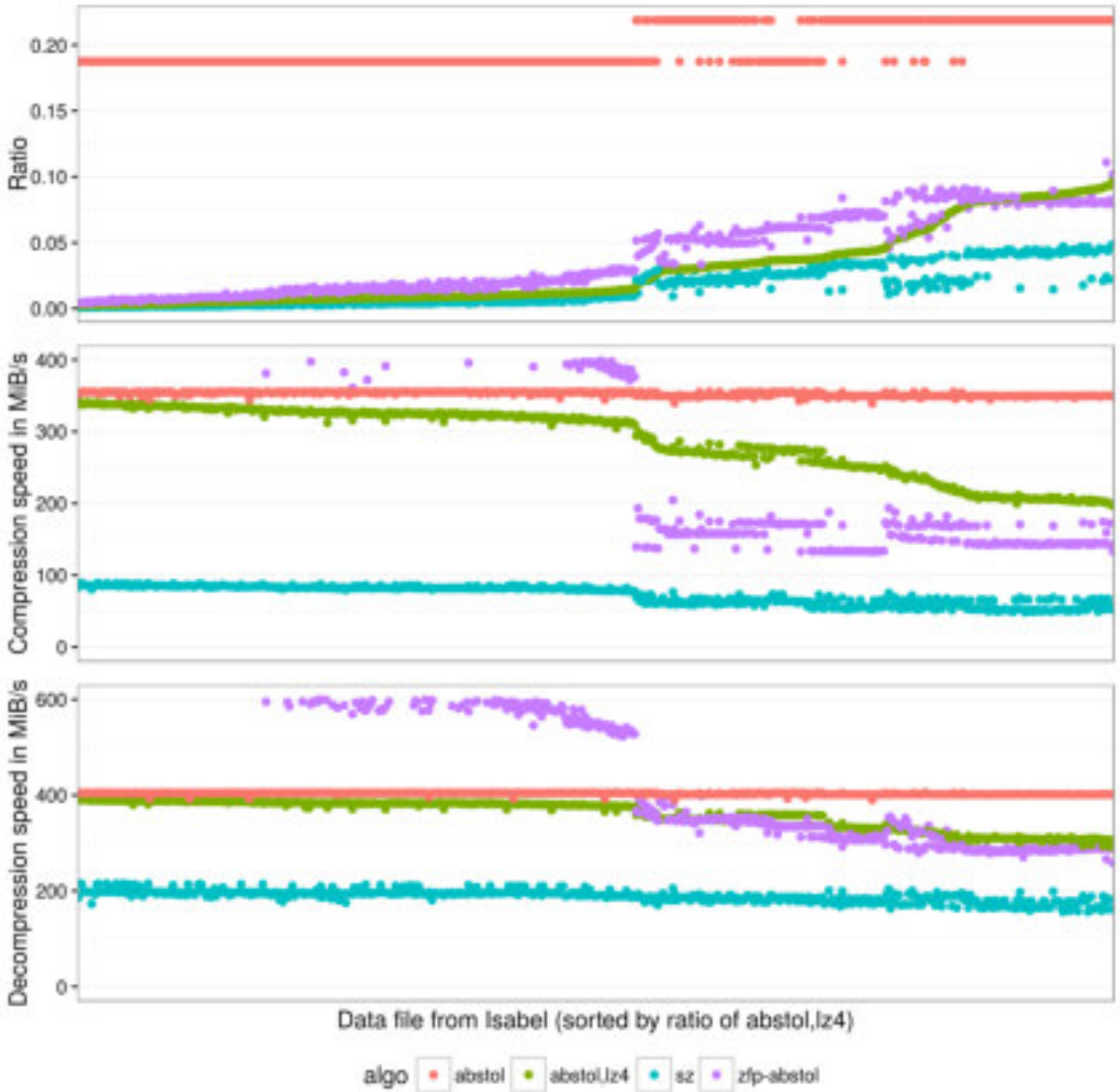
Recap: Goals of the WP2

- 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

- Previous achievements (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)
- Recent progress (second year)
  - Survey of file formats
  - Refactoring (Project structure, quantities)
  - New tool: Pattern creator
  - HDF5 plug-in
  - Evaluation on synthetic data patterns
  - Evaluation on scientific data (hurricane model Isabel)

WP2: Results for Absolute Tolerance of Isabel



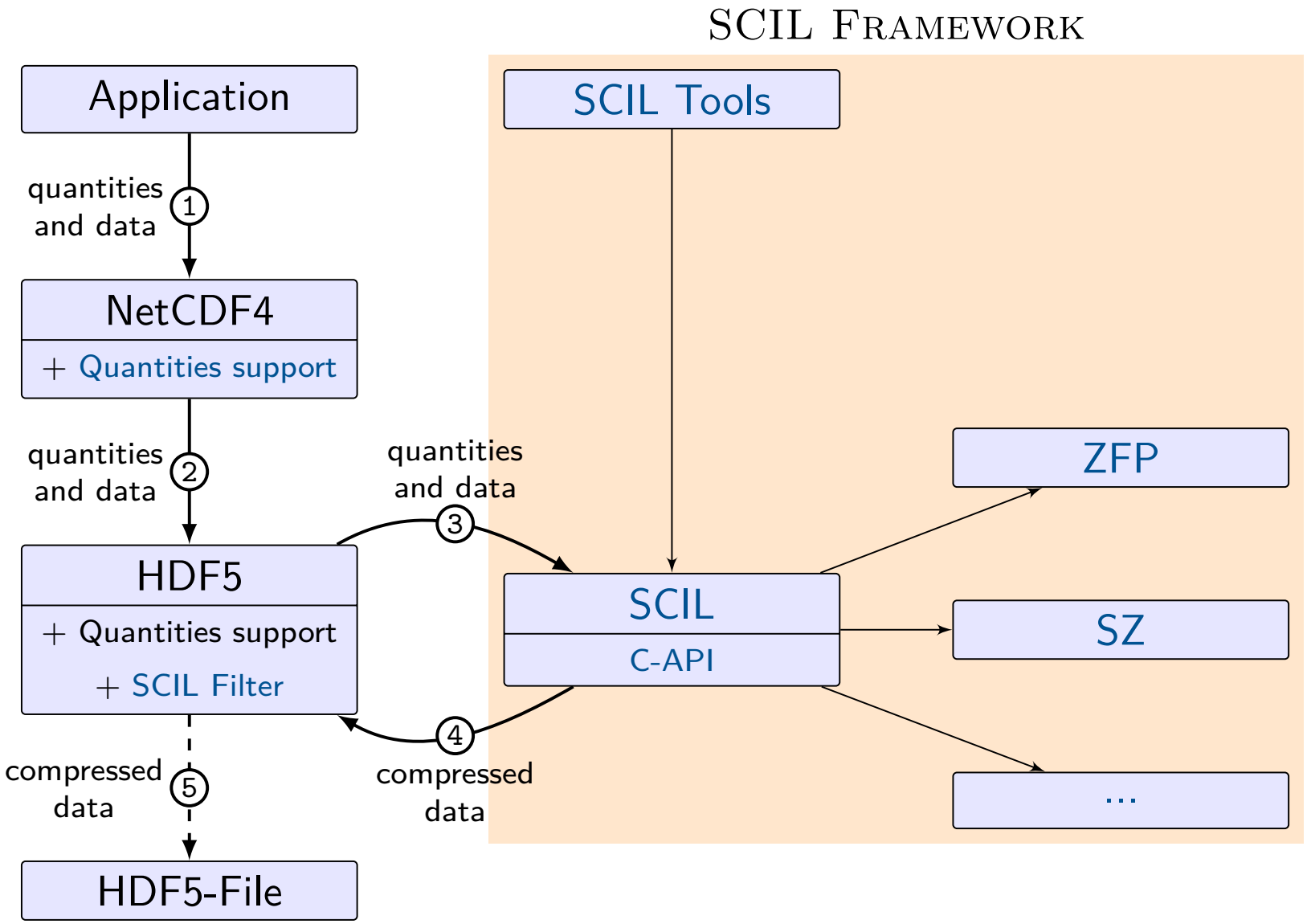
Data

- Hurricane model Isabel
- Single precision (1+8+23 bits)
- 624 variables (100 MB pro var)

Experiments

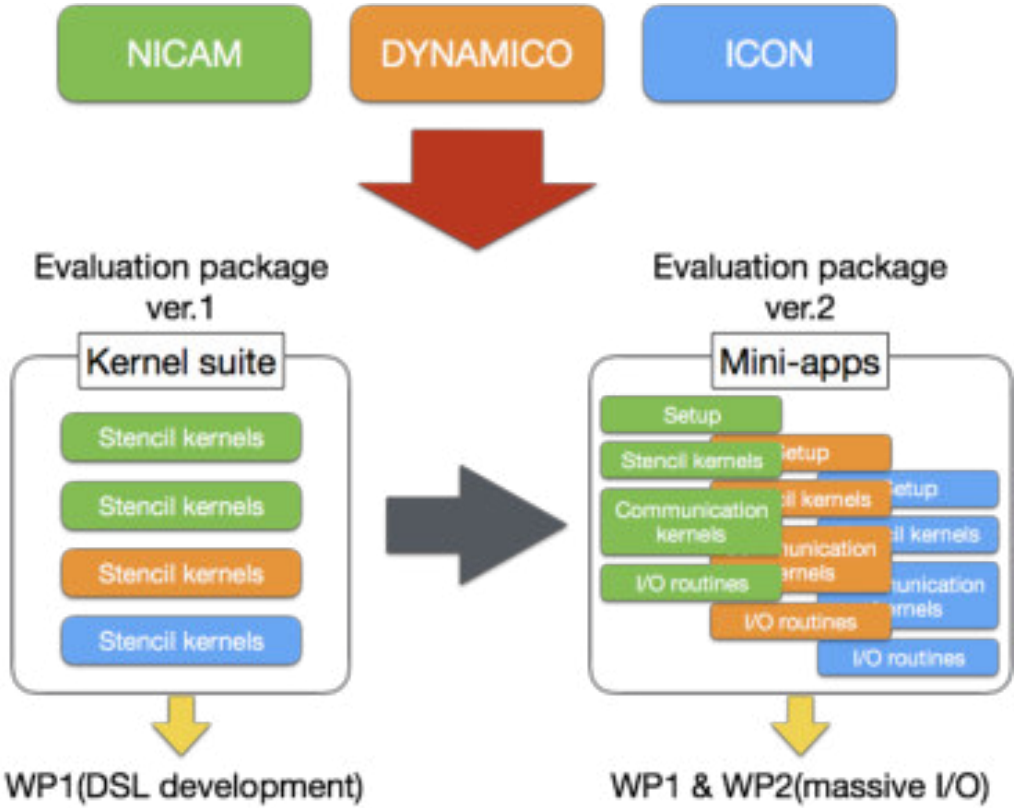
- Test system: Intel i7-6700 CPU (Skylake) with 4 cores @ 3.40GHz
- Single thread, 10 repeats
- Lossy compression with absolute tolerance of 1% of the maximum value (10%, 2%, 1%, 0.2%, 0.1%)

WP2: HDF5 Plug-in



WP 3: Evaluation

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



Progress

- Previous achievements (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
- Recent progress (second year)
  - Packaging IcoAtmosBenchmark v.1
  - NICAM kernels
    - Documents
    - Implementation by GridTools
  - DYNAMICO kernels
    - Extraction, Performance check, Documents

Kernel Extraction

- Extracted kernels from NICAM
  - stencil kernels on the structured grid
    - 2-D(horizontal) diffusion: simple stencil
    - 1-D(vertical) tridiagonal matrix solver: with (ij,k) array, recurrence k-axis
    - 3-D divergence damping: simple but large memory footprint
    - 2-D(horizontal) flux calculation with remapping: large memory footprint
    - 2-D(horizontal) flux limiter for tracer transport: complex, max()/min()
    - 1-D(vertical) flux limiter for tracer transport: complex, max()/min()
    - Setup routine for the coefficients of the stencil operators
  - Communication kernel: unstructured node topology
- Extracted kernels from DYNAMICO
  - stencil kernels on the structured grid
    - potential vorticity calculation
    - geopotential calculation
    - horizontal solver
    - vertical solver
- Extracted kernels from ICON
  - stencil kernels on unstructured grid

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 implementation of the translation tool has been improved
  - Multi-core and GPUs are now supported
    - On single-node and multi-node configurations
- AIMES addresses domain-specific lossy compression
  - (Help) scientists to define the variable accuracy
  - Exploit this knowledge in the compression scheme
  - Novel schemes compete with exististing algorithms
    - The choosing algorithm should always pick the best

Next Steps

- Investigate further inter-kernel optimization opportunities within the DSL translation process
- Investigate IO improvements with the DSL
- Provide shared DSL conventions and survey more scientists
- Extract and convert mini-apps of models to DSL
- Implement SCIL-compression in NICAM
- Evaluate algorithm covering all accuracy quantities

Backup

Backup

Differences among three icosahedral atmospheric models

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

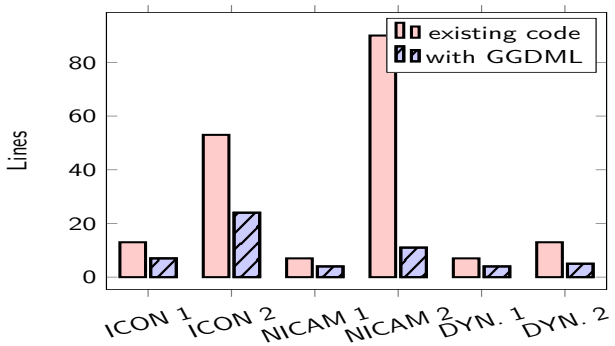
  - semi-structured means... "structured for stencil operation, unstructured for communication topology"

GGDML Impact on the Source Code

The DSL reduces development and maintenance effort

- LOC statistics

Model, kernel	lines (LOC)		words		characters	
	before DSL	with DSL	before DSL	with DSL	before DSL	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

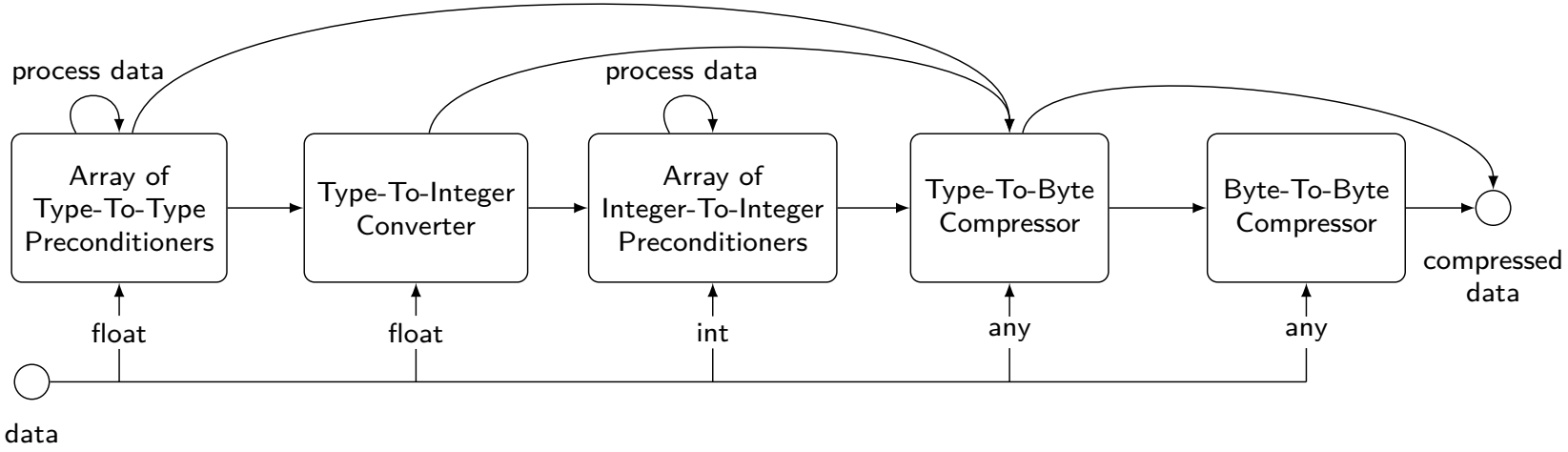
Cooperation Partners

- 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 $\pm$ 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:

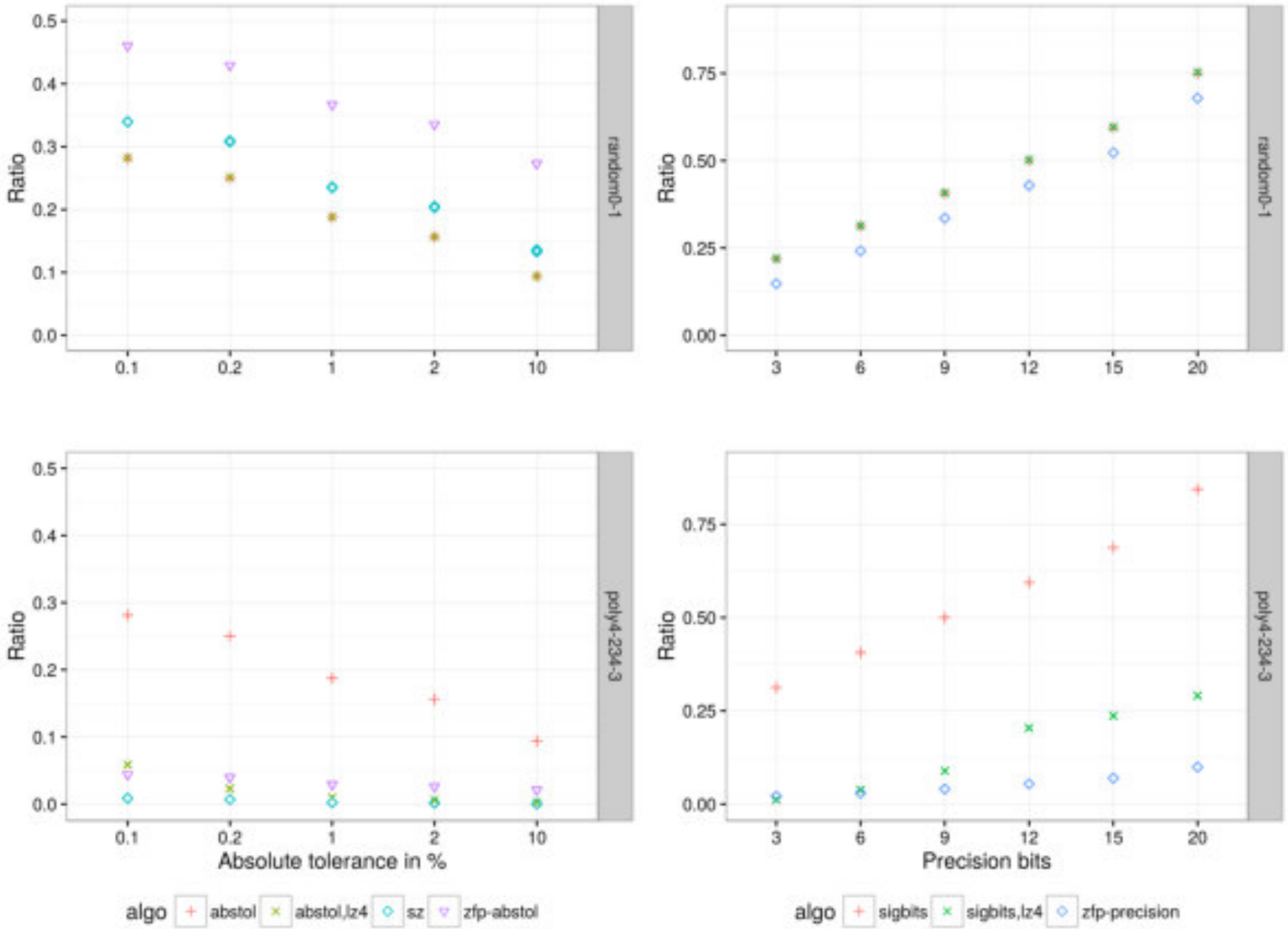
compression speed: in MiB or GiB, or relative to network or storage speed

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

Supplementary quantities:

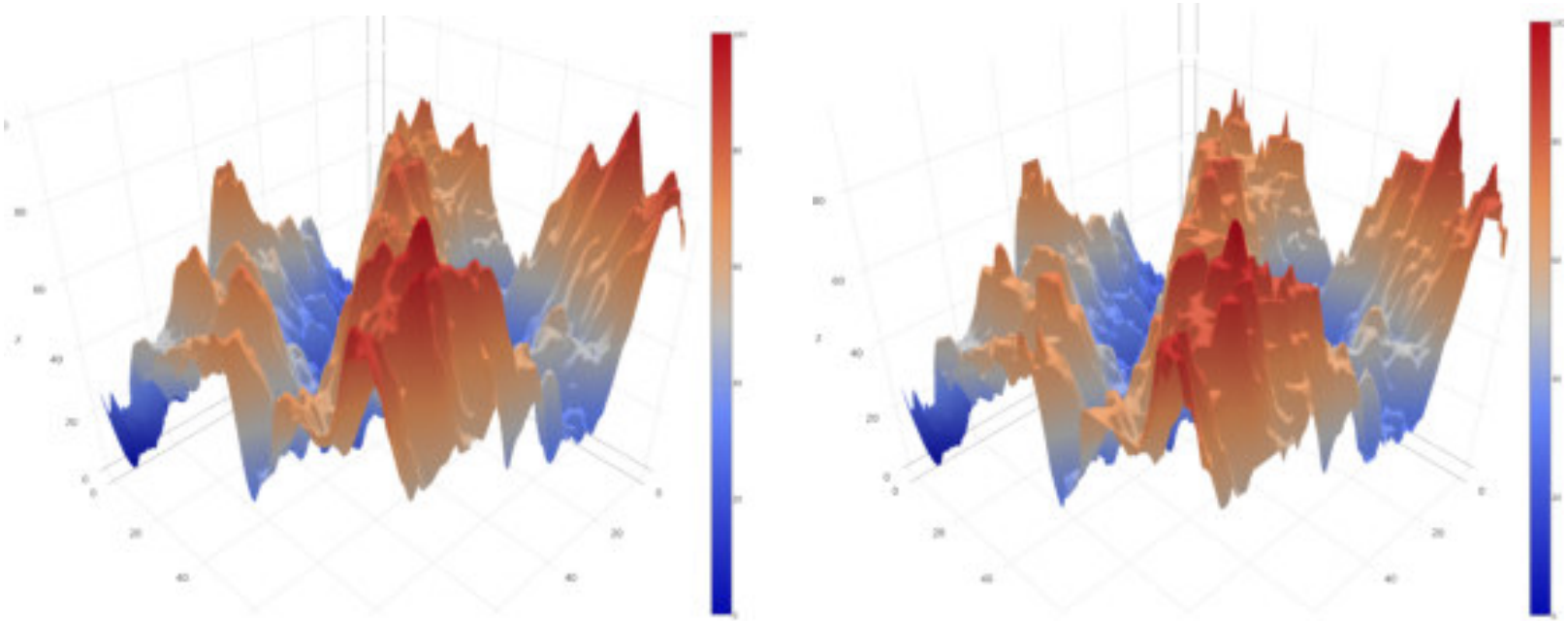
fill value: a value that scientists use to mark special data point

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