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

<table>
<thead>
<tr>
<th></th>
<th>Serial</th>
<th>P100</th>
<th>V100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D</td>
<td>1.97</td>
<td>220.38</td>
<td>91.34</td>
</tr>
<tr>
<td>3D-1D</td>
<td>1.99</td>
<td>408.15</td>
<td>38.75</td>
</tr>
</tbody>
</table>
Performance Evaluation

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

SCIL Framework

1. Application
   quantities and data
   NetCDF4
   + Quantities support
   2. quantities and data
   HDF5
   + Quantities support
   + SCIL Filter
   3. quantities and data
   SCIL
   C-API
   ZFP
   4. compressed data
   compressed data
   .....
   5. HDF5-File
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 existing 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
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**

<table>
<thead>
<tr>
<th>Model, kernel</th>
<th>lines (LOC)</th>
<th>words</th>
<th>characters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>before DSL</td>
<td>with DSL</td>
<td>before DSL</td>
</tr>
<tr>
<td>ICON 1</td>
<td>13</td>
<td>7</td>
<td>238</td>
</tr>
<tr>
<td>ICON 2</td>
<td>53</td>
<td>24</td>
<td>163</td>
</tr>
<tr>
<td>NICAM 1</td>
<td>7</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>NICAM 2</td>
<td>90</td>
<td>11</td>
<td>344</td>
</tr>
<tr>
<td>DYNAMICO 1</td>
<td>7</td>
<td>4</td>
<td>96</td>
</tr>
<tr>
<td>DYNAMICO 2</td>
<td>13</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>total</td>
<td>183</td>
<td>55</td>
<td>911</td>
</tr>
<tr>
<td>relative size with dsl</td>
<td>30%</td>
<td>47%</td>
<td>45%</td>
</tr>
</tbody>
</table>

- **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

<table>
<thead>
<tr>
<th>Software project</th>
<th>Version</th>
<th>Effort Applied</th>
<th>Dev. Time (months)</th>
<th>People require</th>
<th>dev. costs (M€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semi-detached</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSL</td>
<td>2462</td>
<td>38.5</td>
<td>64</td>
<td>12.3</td>
</tr>
<tr>
<td>Organic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSL</td>
<td>1133</td>
<td>29.3</td>
<td>39</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSL</td>
<td>1295</td>
<td>38.1</td>
<td>34</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSL</td>
<td>625</td>
<td>28.9</td>
<td>22</td>
<td>3.1</td>
</tr>
</tbody>
</table>
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:

![Diagram of architecture](attachment:image.png)
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 defining 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 (memcpy 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

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

(a) 1% absolute tolerance

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

(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