natESM Sprint on Vectorization of ICON Code to improve performance

Sergey Sukov[2], **Trang Van Pham**[1] , **Jens-Olaf Beismann**[3], Kristina Fröhlich[1]

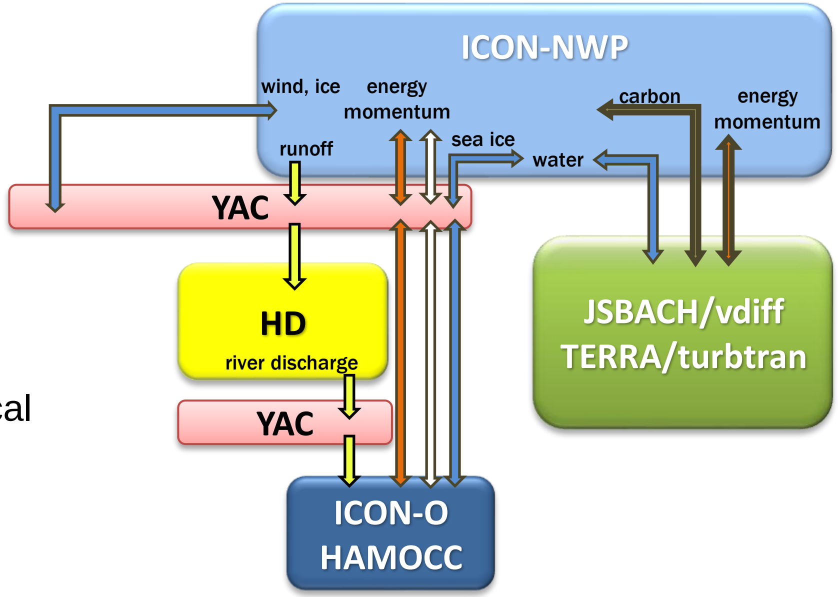
[1] Deutscher Wetterdienst, Offenbach, Germany
[2] Jülich Supercomputing Center, Forschungszentrum Jülich GmbH, Jülich, Germany
[3] NEC Deutschland GmbH, Düsseldorf, Germany

Motivation of the Sprint

***ICON-XPP**: eXtended Prediction and Projection*

- ✓ Applications at DWD:
  - ✓ climate projections in CMIP framework
  - ✓ climate forecasts on seasonal/decadal scales

- ✓ Fully coupled atmosphere-land-ocean-river-carbon-cycle:
  - ✓ Atmosphere ICON-NWP r2b5/80km
  - ✓ Land ICON-Land (JSBACH/Quincy)
  - ✓ Ocean ICON-O r2b7/20km with active bio-geochemical component HAMOCC
  - ✓ Hydrological model HD (latlon, 1 deg)
  - ✓ Externally coupled via YAC



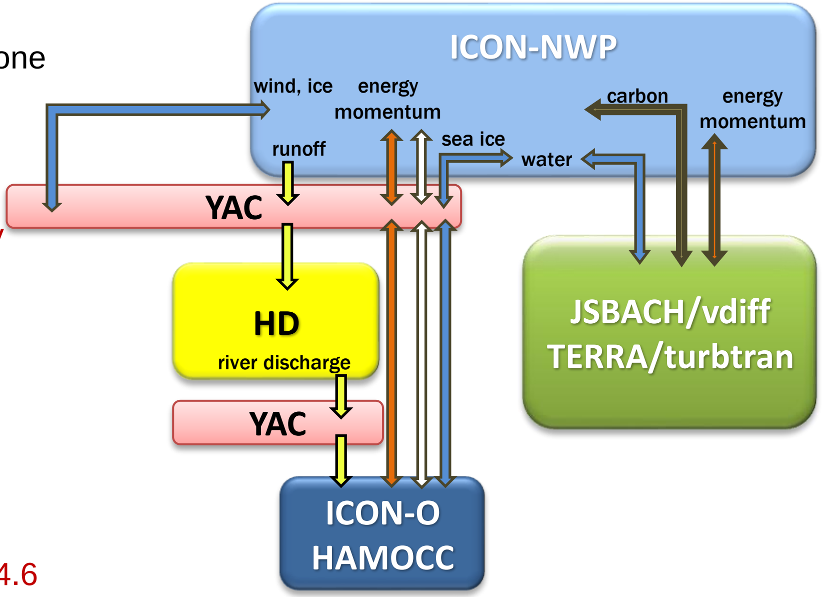
Motivation of the Sprint

ICON-XPP: *eXtended Prediction and Projection*

Within the BMBF CAP7 project:

- ✓ Climate projection runs for CMIP7 ~34.000 model years should be done within 2026
- ✓ Computational time on Levante is sparse because of high demands from other projects
- ✓ Slowest components are ICON-O and HAMOCC (which is especially needed for the emission driven runs)

- ✓ Performance on different architectures:
  - ✓ DKRZ/Levante (CPU machines): 206 nodes, 30 SYPD
  - ✓ ECMWF/Atos (similar architechture like Levante but older model): same settings with 206 nodes, 20 SYPD (no further setting optimization)
  - ✓ DWD/RCL (vector machine): 20 nodes (160 Vector engines), 4.6 SYPD



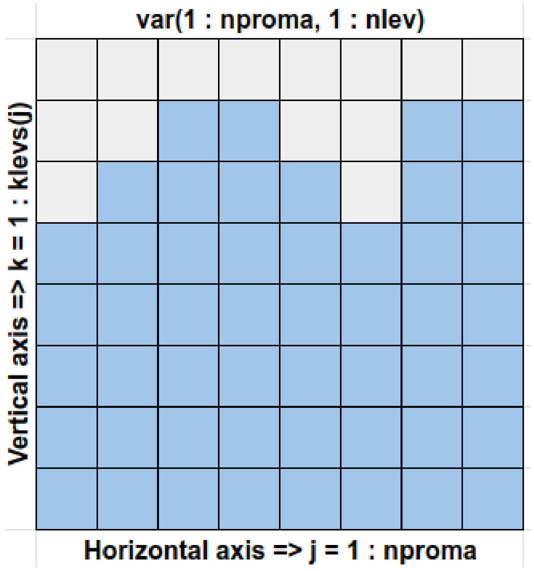
ICON-Ocean vectorization

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

- HPC devices (GPUs, Vector processors) exploit **data parallelism**.
- **Long inner loops** and **contiguous memory access** are necessary for efficient use of vector processors.
- Parts of ICON-Ocean have been amended correspondingly (exchange of **loop order**), currently two code versions exists.
- Ongoing code review aiming for a **single code version** using directives for different platforms (collaboration with D. Zobel regarding GPUs, DKRZ).
- Modules like HAMOCC (including iterative processes, written with „grid point focus“) require a different approach and potentially more extensive code reorganization. => main task for Sergey

1. Classical Loop Reordering

Deutscher Wetterdienst
Wetter und Klima aus einer Hand


var(1 : nproma, 1 : nlev)
Vertical axis => k = 1 : klevs(j)
Horizontal axis => j = 1 : nproma

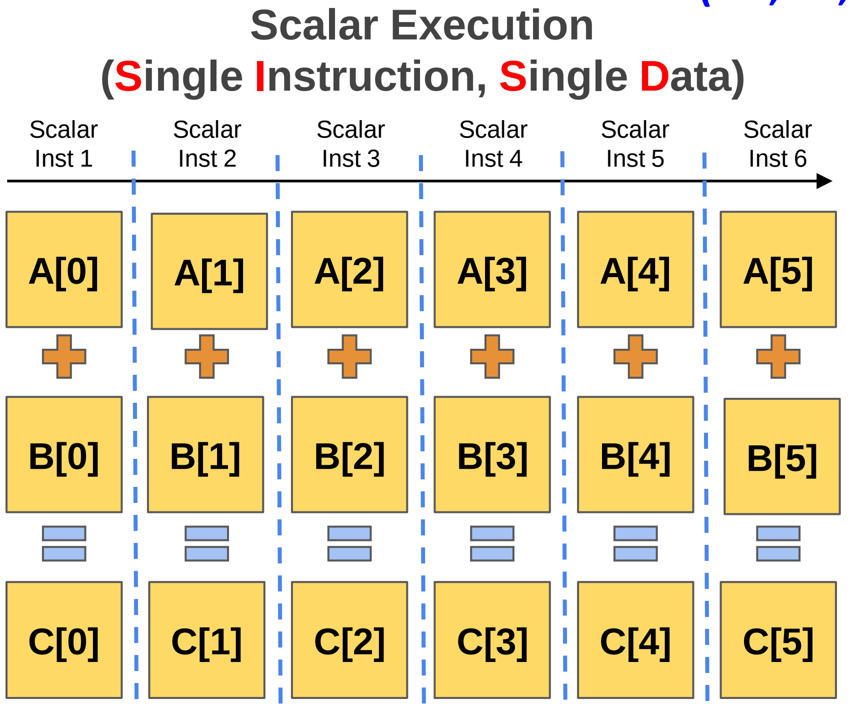
`DO j = 1, nproma`
`   DO k = 1, klev(j)`
`      var(j,k) = ...`
	Original code
	● Access pattern does not match memory layout
	● GPU thread load imbalance

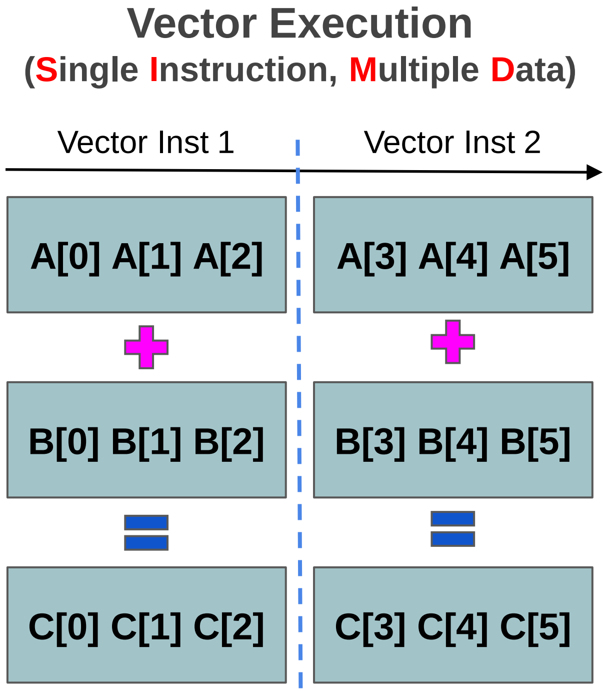
`klevMAX = MAXVAL(klev(1:nproma))`
`DO k = 1, klevMAX`
`   DO j = 1, nproma`
`      IF(k <= klev(j)) THEN`
`         var(j,k) = ...`
	Reordering loops
	● Collapsed loops and coalesced global memory access on GPU
	● Inactive iterations in the loop

1. Vector Computations (Loop Vectorization)

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

for(i=0;i<6;i++) A[i]=B[i]+C[i]





SSE / AVX / AVX-512

Techniques Used For Code Vectorization

TECHNIQUE	REASON / IMPACT ON CODE STRUCTURE
Reordering Loops	- Unit-Stride Memory Access - Full Utilization of Large Vector Registers (NEC SX-Aurora)
Inlining / Vectorization Compiler Pragmas	- Hints that the Loop Iterations are Independent - Subroutine Inlining
Loop Splitting	- Replacing Scalar Local Variables by Vectors - Extending Temporary Arrays with an Additional Dimension
Replacing Scalar Functions and Subroutines with Array-Based Procedures	- Additional Temporary Arrays for Storing Input Parameters - Dual Implementations of Essentially the Same Procedure
Implementing a Vectorization Mask	Extra Boolean Array

Main Constraints:

- Vectorization is applied only to the innermost loop
- Function calls inside the vectorized code block are not supported

🟩 **Minor code restructuring**

🟥 **Major code restructuring**

Vectorization of Key HAMOCC Subroutines

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Vectorization Techniques: Loop reordering and splitting; Replacing scalar functions with vector equivalents

Subroutine	Execution time (seconds)		Speedup
	Before vectorization	After vectorization	
calc_dissol	**216.36**	**4.43**	48.8
gasex	4.78	0.13	38.2
powach	**36.42**	**0.96**	38.1
ocprod	**159.98**	**2.66**	60.2
mean_agg_sink_speed	**442.24**	**8.28**	53.4
chemcon	4.76	0.07	70.0
settling	**17.23**	**0.23**	76.6
cyadyn	4.43	1.21	3.7
update_bgc	3.20	0.57	5.7
swr_absorption	1.96	0.29	6.8
update_icon	1.78	0.27	6.6
set_bgc_tend_output	0.78	0.78	1.0
TOTAL	**893.92**	**19.85**	**45.0**

Achieved Performance of Coupled Runs

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Number of vector engines	Number of MPI processes		SYPD
	Ocean	Atmosphere	
64	352	159	9.7
80	464	175	11.4
128	752	271	15.2
160	**976**	**303**	**17.0**
160	1026	263	4.6 (before the Sprint)

Performance Constraints

- ➔ Load imbalance
- ➔ Nonlinear runtime scaling with problem size
- ➔ Coupling overhead between componentes

The report can be found here:

https://www.nat-esm.de/services/support-through-sprints/documentation/natesm_sprint_report_icon-xpp_opt_fv.pdf

This work is now available in the ICON Open Source Release:

https://gitlab.dkrz.de/icon/icon-model/-/tags/icon-2025.10-2-public

Many thanks to
Sergey, Iris (natESM), Jens-Olaf (Nec)
and others