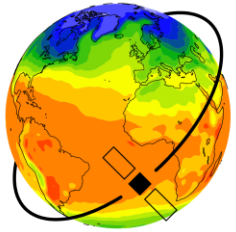
natESM



ESMValTool

Earth System Model Evaluation Tool

DKRZ DEUTSCHES KLIMARECHENZENTRUM

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

DKK Deutsches Klima Konsortium

Federal Ministry of Education and Research

Sprint 6

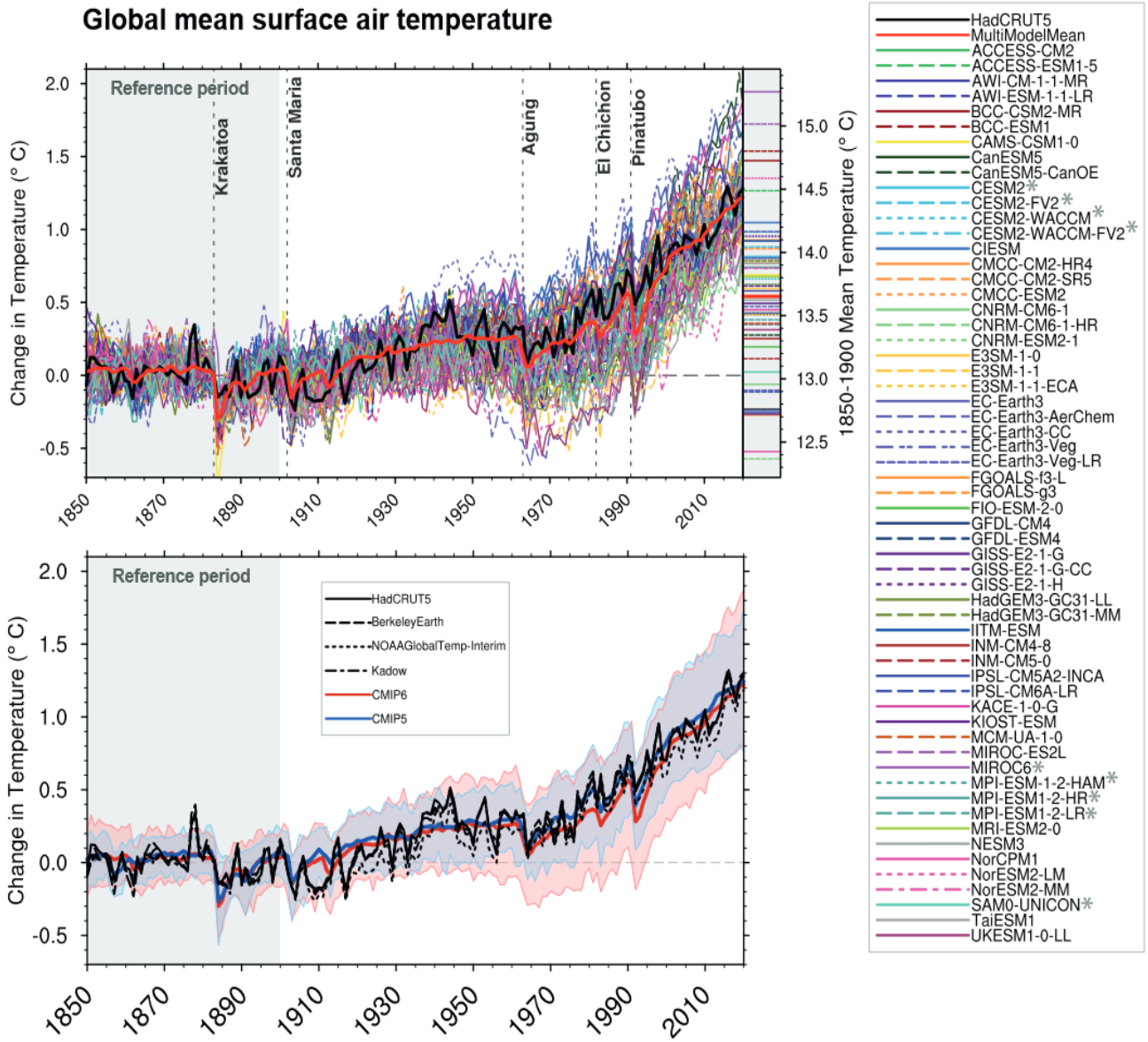
Challenges and results experienced during the ESMValTool sprint

Birgit Hassler (DLR), **Rémi Kazeroni** (DLR)*, **Manuel Schlund** (DLR), **Jörg Benke** (JSC)

* now at CNRS-IPSL

General information - I

- community-developed, open-source software tool for the evaluation and analysis of output from ESMs
- well-documented source code, scientific background documentation, as well as a detailed description of the technical infrastructure
- provenance record that allows for traceability and reproducibility
- has been used for contributions to several chapters of the Sixth Assessment Report (AR6) of the Intergovernmental Panel on Climate Change (IPCC)



Observed and simulated time series of the anomalies in annual and global mean near-surface air temperature (Figure 3.4 from Eyring et al., 2021).

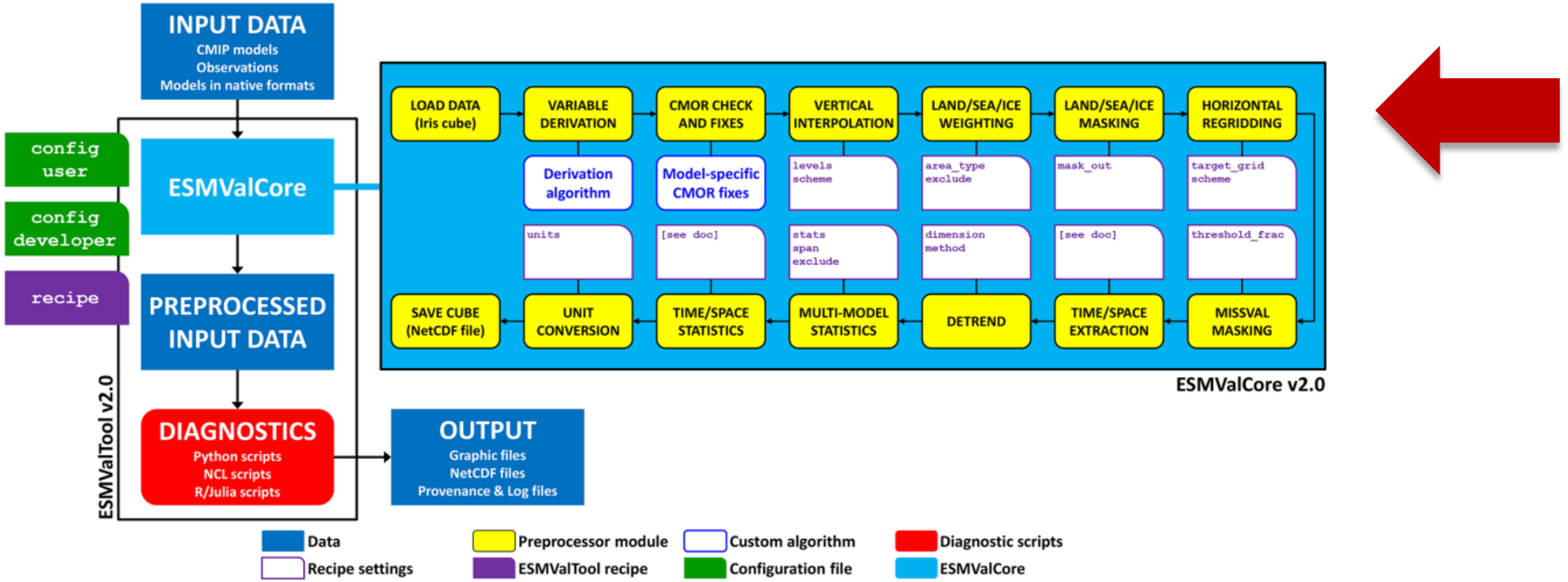
General information - II

- *GitHub:* https://github.com/ESMValGroup/ESMValTool and https://github.com/ESMValGroup/ESMValCore
- *Online documentation (Sphinx and ReadTheDocs):* https://docs.esmvaltool.org/
- *Tutorial:* https://esmvalgroup.github.io/ESMValTool_Tutorial/
- *Website*: https://www.esmvaltool.org/
- *Usage:* used in ~60 peer-reviewed articles
- *License:* Apache-License, Version 2.0
- *ESMValCore language:* Python 3
- *ESMValTool supported languages:* Python 3, NCL, R, Julia
- *doi:* 10.5281/zenodo.3387139 (ESMValCore) and 10.5281/zenodo.3401363 (ESMValTool)

Technical and scientific challenges

- new challenges stemming from higher resolution and enhanced complexity (e.g. memory limits): **need to be tackled for ESMValTool to be efficient and scalable in the future, especially to perform multi-model analyses and become exascale-ready**
- preparation for model simulations with a further increased horizontal and vertical resolution, e.g. for the next phase of CMIP or model-intercomparisons for high-resolution cloud resolving models: **efficient memory handling will become crucial**
- processing of native output from the some models (including ICON) for an easier and more convenient way to analyze, evaluate or monitor new simulations is possible, **BUT: for unstructured grids additional information and processing steps are necessary**

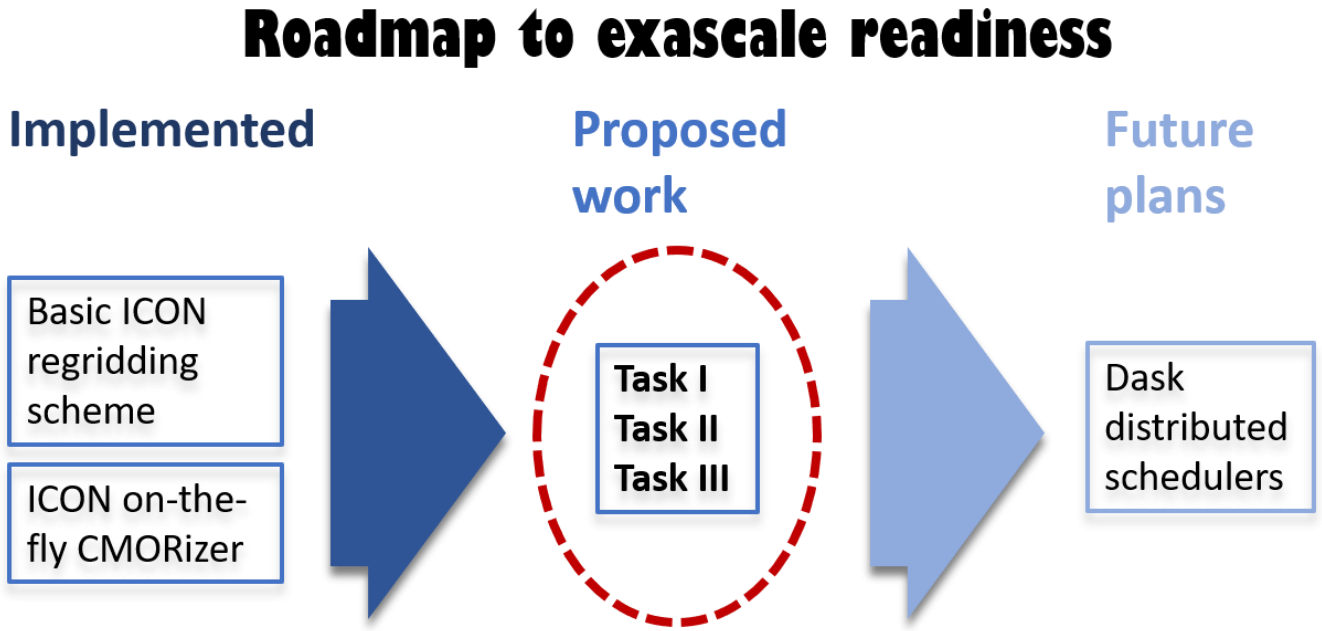
Status before sprint



- Some preprocessor functions are not lazy (~21 out of ~50, at the time of sprint start)
- Requires replacing of NumPy arrays with Dask arrays, or even recoding
- Optimizing dask distributed schedulers for maximum performance enhancement across multiple nodes

ESMValTool sprint

- **Task I**: Optimizing the memory footprint of selected preprocessor functions by taking advantage of Dask
- **Task II**: Developing a concept for using Dask distributed schedulers to further improve memory management
- **Task III**: Extending the support of unstructured grids in ESMValCore by implementing run-time reformatting of input data to the UGRID standard within Iris (e.g. ocean grids)



Challenges

Task	M1	M2	M3	M4	M5	M6
Task 0: Learning about the ESMValCore and ESMValTool and their code structure	X					
Task I: Updating the remaining non-lazy preprocessor functions to be memory efficient		X	X	X	X	
Task II: Studying Dask distributed schedulers to be able to provide advice on how to further improve memory management of ESMValCore						X
Task III: Updating the ESMValCore so that ICON data can be made UGRID-compliant at run time						X

Weekly meetings with natESM RSE and DLR RSE for the duration of the sprint

Challenges

Task	M1	M2	M3	M4	M5	M6
Task 0: Learning about the ESMValCore and ESMValTool and their code structure	X					
Task I: Updating the remaining non-lazy preprocessor functions to be memory efficient		X	X	X	X	
Task II: Studying Dask distributed schedulers to be able to provide advice on how to further improve memory management of ESMValCore						X
Task III: Updating the ESMValCore so that ICON data can be made UGRID-compliant at run time						X

Weekly meetings with natESM RSE and DLR RSE for the duration of the sprint

Challenges

	Task	M1	M2	M3	M4	M5	M6
1	**Task 0**: Learning about the ESMValCore and ESMValTool and their code structure	X					
2	**Task I**: Updating the remaining non-lazy preprocessor functions to be memory efficient		X	X	X	X	
	Task II: Studying Dask distributed schedulers to be able to provide advice on how to further improve memory management of ESMValCore						X
	Task III: Updating the ESMValCore so that ICON data can be made UGRID-compliant at run time						X

Weekly meetings with natESM RSE and DLR RSE for the duration of the sprint

Challenges

	Task	M1	M2	M3	M4	M5	M6
1	**Task 0**: Learning about the ESMValCore and ESMValTool and their code structure	X					
2	**Task I**: Updating the remaining non-lazy preprocessor functions to be memory efficient		X	X	X	X	
3	**Task II**: Studying Dask distributed schedulers to be able to provide advice on how to further improve memory management of ESMValCore						X
X	**Task III**: Updating the ESMValCore so that ICON data can be made UGRID-compliant at run time						X

Weekly meetings with natESM RSE and DLR RSE for the duration of the sprint

Challenges

	Task	M1	M2	M3	M4	M5	M6
1	**Task 0**: Learning about the ESMValCore and ESMValTool and their code structure	X					
2	**Task I**: Updating the remaining non-lazy preprocessor functions to be memory efficient		X	X	X	X	
3	**Task II**: Studying Dask distributed schedulers to be able to provide advice on how to further improve memory management of ESMValCore						X
X	**Task III**: Updating the ESMValCore so that ICON data can be made UGRID-compliant at run time						X

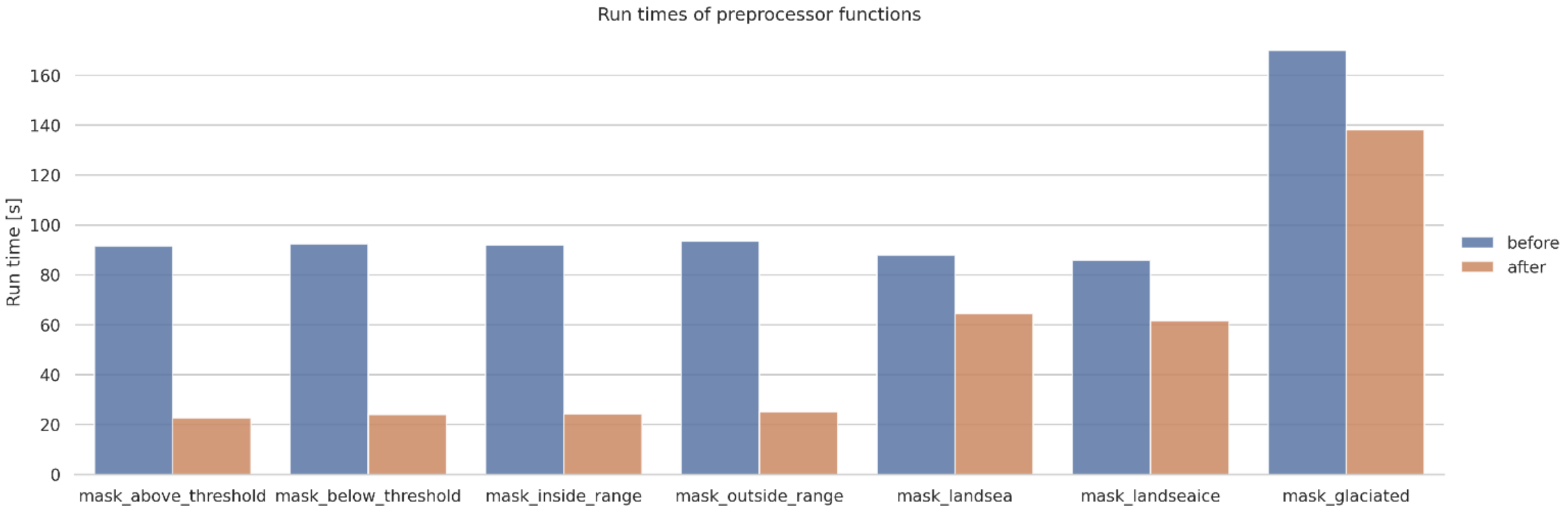
Weekly meetings with natESM RSE and DLR RSE for the duration of the sprint

- Steep learning curve for ESMValCore (needs more than one month time)
- Experience with Dask and Iris would have been helpful
- Additional requests on Jörg's time slowed progress down

Results - I

- **Seven preprocessor** functions were ported to Dask: mask_above_threshold, mask_below_threshold, mask_inside_range, mask_outside_range, mask_landsea, mask_landseaice and mask_glaciated

Results - II



Runtimes of different ESMValTool preprocessor functions before (blue) and after (orange) making the corresponding function lazy. In this example, one single preprocessor function is applied to 65 years of daily 3D data of the CMIP6 model CanESM5. The tests were performed on a single compute node on DKRZ's Levante with 128 cores and 256 GB of RAM.

Results - I

- **Seven preprocessor** functions were ported to Dask: mask_above_threshold, mask_below_threshold, mask_inside_range, mask_outside_range, mask_landsea, mask_landseaice and mask_glaciated
- Clear **reduction in runtime** (between 20-25% and 80%)
- ESMValTool can now also read the data on systems with **less memory than the input data size**. Before these optimizations implemented here, the evaluation would run out of memory.
- **Not all** preprocessor functions have been ported yet
- Optimizing Dask distributed schedulers was started but **not fully explored** so that no further runtime improvement was achieved

Outlook & open questions

- **Additional sprint** was granted within the European project **ESiWACE phase 3** (https://www.esiwace.eu/) where more of the non-lazy preprocessor functions will be ported (by now more than 50 preprocessor functions are available for ESMValCore, of which about half is in lazy format)
- Within the ESMValTool community, the capability of converting data at run time to the **UGRID standard** has been implemented now
- **Dask distributed schedulers** are optimized by ESMValTool technical core developers over time
- Reading ICON data on-the-fly is now being expanded to reading also **ICON-Seamless data**