Contribution ID: **8** Type: **Poster**

CGDycore: A Julia implementation of numerical dycores for different backends

Tuesday 8 October 2024 13:54 (6 minutes)

Modern computer systems are more and more diverse and composed of nodes with shared memory and different type of processors, mainly CPU's and GPU's. GPU vendors are coming usually with their own language specification like CUDA (Nvidia), ROCm (AMD), or Metal (Mac).
The Julia language is designed for scientific computation and comes with special packages to write individual kernels for different backends. KernelAbstractions.jl allows to write typical stencil operations in a native way without a main knowledge about the GPU backend. Combined with domain decomposition and data transfer via MPI-aware implementations from backend to backend results in efficient code for high performance computing.

I will present results for two dycore implementation, a spectral element implementation following the HOMME code (Sandia Lab) and a finite volume implementation following the numerics of the ICON-model in the same Julia infrastructure.

Primary author: OSWALD, Knoth (Institute for Tropospheric Research)

Presenter: OSWALD, Knoth (Institute for Tropospheric Research)

Session Classification: Postersession & Coffee