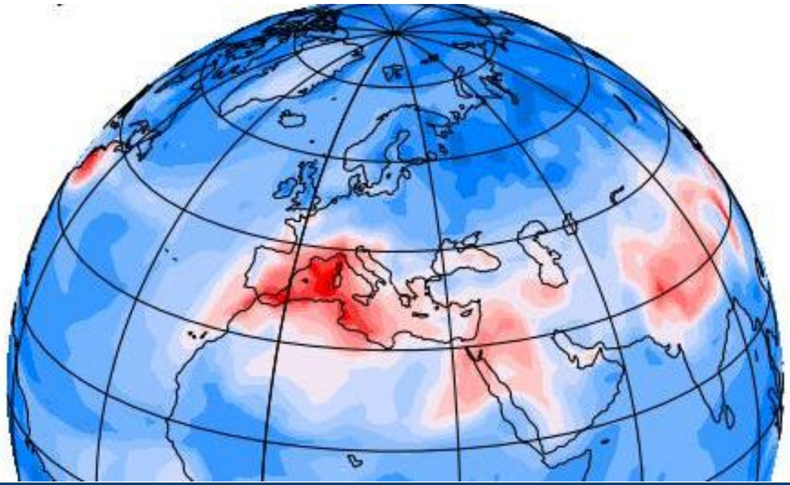


Accelerating chemistry-climate simulations with MESSy

22.05.2022 I KERSTIN HARTUNG (DLR-PA), TIMO KIRFEL (FZJ-IEK8)

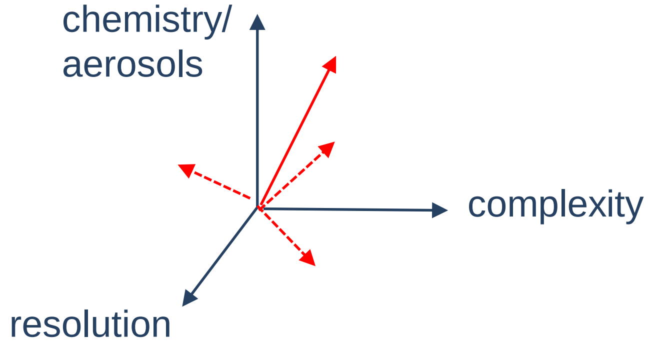
Astrid Kerkweg, Domenico Taraborrelli et al. (FZJ), Patrick Jöckel and Bastian Kern (DLR), participants of the NVIDIA/MESSy meetings

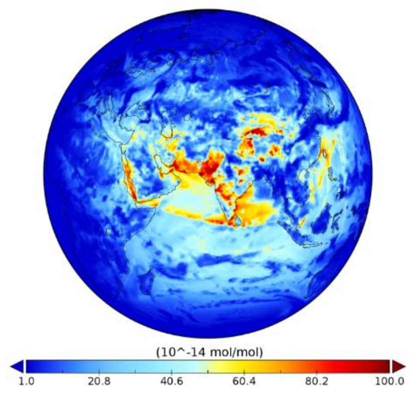
Thanks to JSC‘s NVIDIA Application Lab

MESSY

The Modular Earth Submodel System

- **MESSy** has more than 200 users as part of a large international consortium
- **MESSy** allows to run chemistry-climate model simulations of various complexity, e.g. as idealized simulations or with advanced multi-phase chemistry
- **MESSy** is e.g. used as part of the Chemistry Climate Model Intercomparison (CCMI) project(s) [1], for health and hazard studies (e.g. [2] and [3]) and to complement measurement campaigns (e.g. [4] and [5])
- **MESSy's** infrastructure and scientific description are continuously developed => **e.g. porting to GPUs**



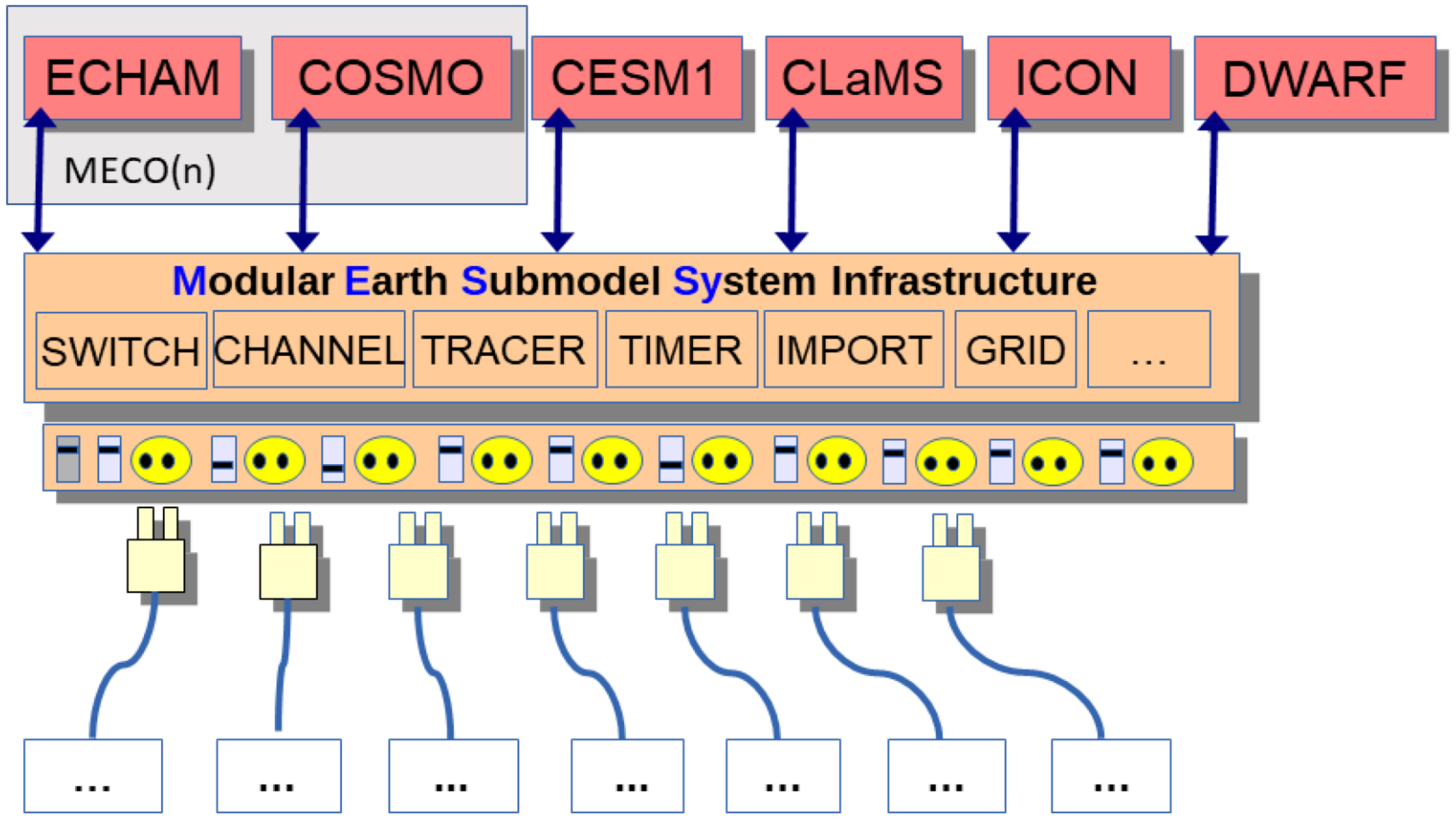


Hydroxyl (OH) at ground level

T255L31, 60 km / 0.5°
Domenico Taraborrelli (FZJ)

MESSY CODE STRUCTURE

The MESSy basemodel family



Legacy models

MESSy infrastructure: „middleware“ for „interoperability“

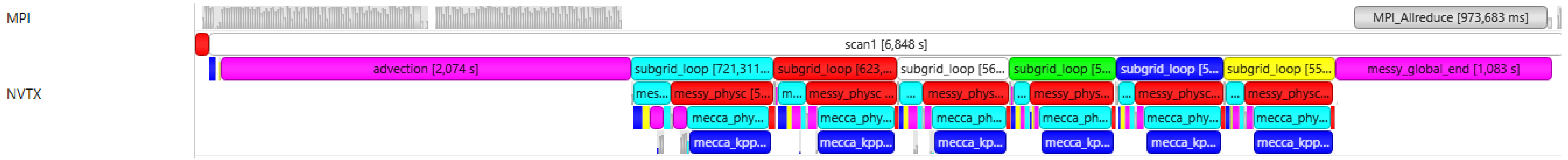
MESSy submodels: chemistry, physics, diagnostics,…

EMAC PROFILE

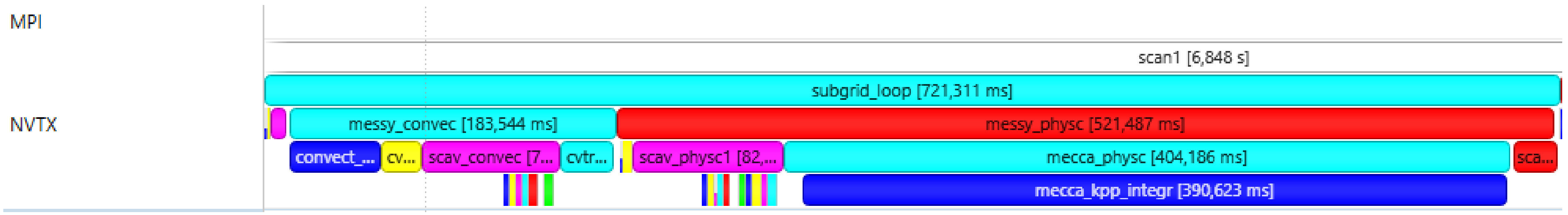
CCMI2

- Chemistry–Climate Model Initiative (CCMI2) setup with around 160 chemical species
- MPI parallelization (2 nodes) and additional spatial decomposition (“vector blocking”)

Full time step



One grid-box loop



EMAC PROFILE

CCMI2

submodel or marker	description	using KPP	% of subgrid_loop
MECCA	gas phase chemistry	yes	67.9
SCAV	liquid/ ice phase chemistry & wet deposition	yes	16.0
CONVECT	convection		4.8
CVTRANS	convection tracer transport		4.8
radiation	radiation		1.4
vdiff	vertical turbulent exchange		1.2
MSBM	multiphase stratospheric box model		1.0
JVAL	photolysis rates		0.6

MECCA - KPP

Module **E**fficiently **C**alculating the **C**hemistry of the **A**tmosphere **- K**inetic **P**re**P**rocessor

MECCA	An atmospheric chemistry module based on a comprehensive chemical mechanism with tropospheric and stratospheric chemistry of both the gas and the aqueous phases [6].

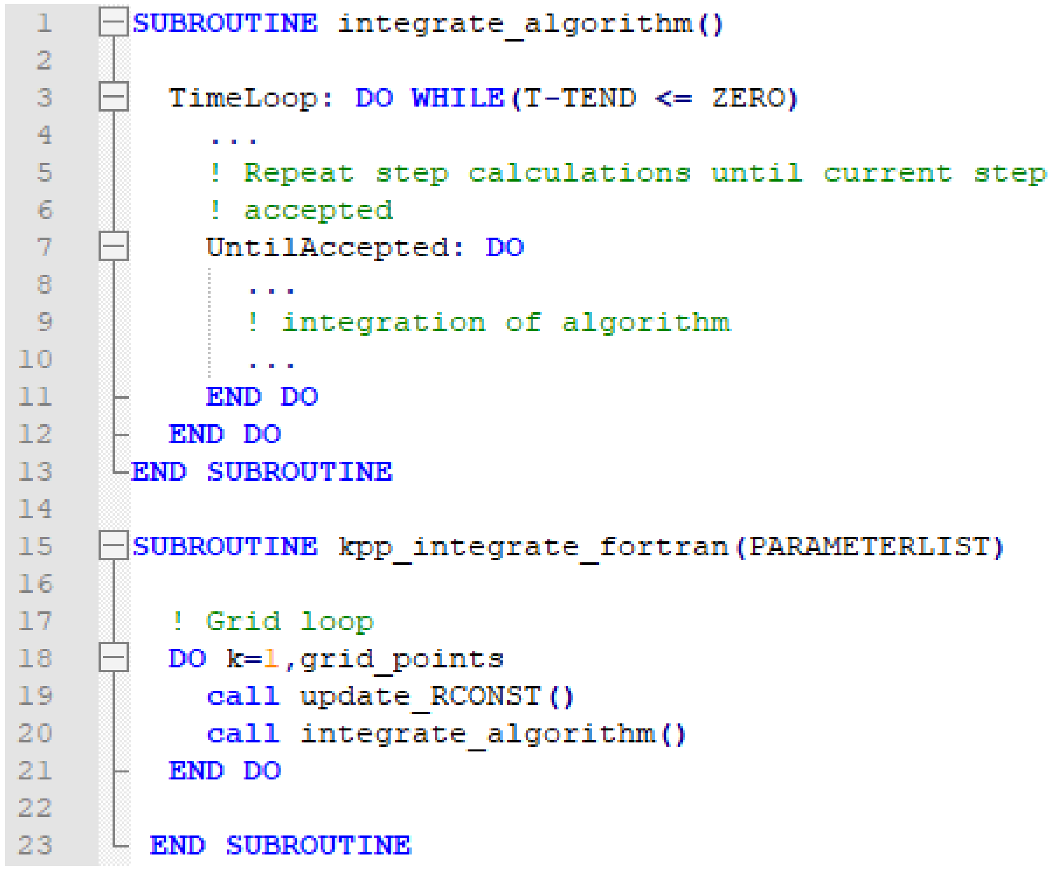
Uses KPP software for solving the stiff chemical ODE systems [7]

KPP	A software tool that assists the computer simulation of chemical kinetic systems.

KPP AND ATMOSPHERIC CHEMISTRY

Overview of the kinetic preprocessor

- problem described by reaction rates and chemical concentrations
- integration is done per grid point (grid loop)
- here: Rosenbrock 3 method with adaptive time stepping to solve the system of stiff ODEs
- determine function and its Jacobian each iteration step



```fortran
SUBROUTINE integrate_algorithm()

  TimeLoop: DO WHILE(T-TEND <= ZERO)
    ...
    ! Repeat step calculations until current step
    ! accepted
    UntilAccepted: DO
      ...
      ! integration of algorithm
      ...
    END DO
  END DO
END SUBROUTINE

SUBROUTINE kpp_integrate_fortran(PARAMETERLIST)

  ! Grid loop
  DO k=1,grid_points
    call update_RCONST()
    call integrate_algorithm()
  END DO

 END SUBROUTINE
```

MECCA – KPP: PORTING EFFORTS

CUDA implementation

- MECCA KPP was first part of MESSy to be ported to GPU
- Done by T. Christoudias (from Cyprus Institut) et al. (e.g. [3])

- implementation facts
  - a Fortran source code to CUDA source code parser written in Python called Medina
  - GPU threads are used to solve the ODE system of every grid point in parallel (1 grid point -> 1 thread)
  - some adaptations were required to run on JUWELS GPUs: in original approach too much GPU RAM used

CUDA IMPLEMENTATION

RAM usage

Issue: first implementation uses too much thread-local memory on the GPU for allocating arrays

Solutions:

1. on newer GPUs (Volta) the usage of CUDA-MPS with the parameter CUDA_ACTIV_THREAD_PERCENTAGE limits the number of threads a task can use
   - ➢ does not work for mechanisms with more than 600 species and for older GPUs
2. allocate the arrays normally in the global GPU memory (and adapt parser)
   - ➢ this reduces the amount of allocated thread-local memory and the RAM usage
   - ➢ the performance decreases around 15%

CUDA IMPLEMENTATION

Results

- 3-month T63L90 simulation on 8 JUWELS Booster nodes, with CUDA_MPS and CCMI2 as chemical mechanism

	CPU	GPU local mem	GPU global mem
Runtime (h)	16.09	10.08	10.31
Speedup to CPU		1.60	1.56
GPU RAM usage (GB)		11.1	7.1

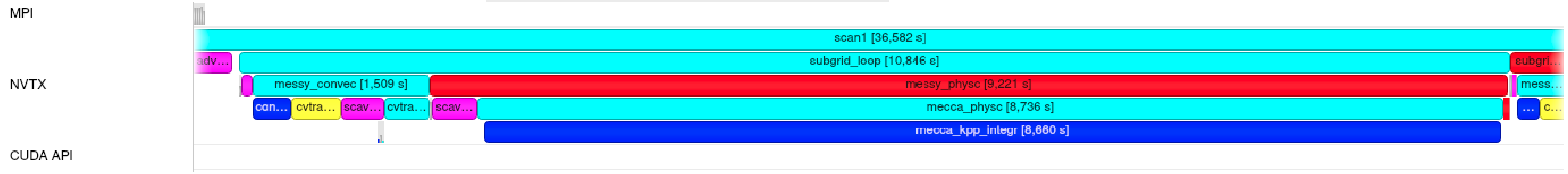
- good stability of solver in long-term simulation (10 years)

CUDA IMPLEMENTATION

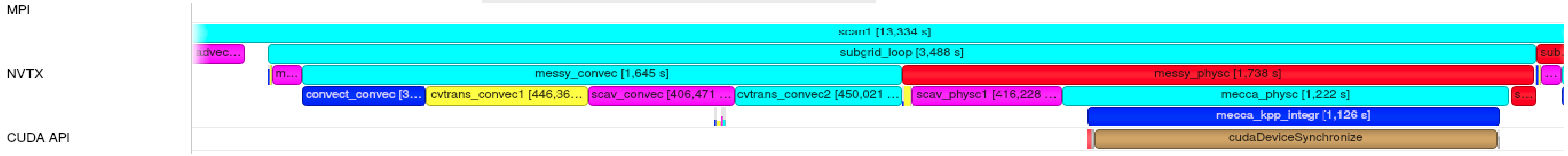
Results (T42L90 with MOM chemistry, 666 species)

MOM setup: out-of-memory with the GPU local variant, needs about 18 GB with the modified approach, overall speedup: 2.74x

CPU



GPU



CUDA IMPLEMENTATION

Outlook

- Second most expensive part is the SCAV module
  - Also partly also generated by KPP
  - First prototype version was finished most recently: very small change in speedup
- GMXe (calculating aerosol phase chemistry) is a third module which uses KPP and can be adapted similarly

OpenACC AND MECCA KPP

Implementation

Use OpenACC instead of CUDA for accelerating MECCA/KPP on GPUs:

- to avoid extra CUDA file and extra parser for generating the code
- to allow for an easier and modifiable transition of KPP development to submodules SCAV and GMXe, which partly use KPP too
- to provide uniform GPU approach when fully porting the model

➡ Get an OpenACC implementation which is as fast as the current CUDA one

OpenACC IN MECCA KPP

Implementation

Get an OpenACC implementation which is as fast as the current CUDA one

Created a `mini app` which can run as standalone executable for testing

Figured out that multiple variants could be successful to accelerate MECCA KPP with OpenACC

OpenACC IN MECCA KPP

Implementation - First variant

– **First variant:** accelerate the outer grid loop

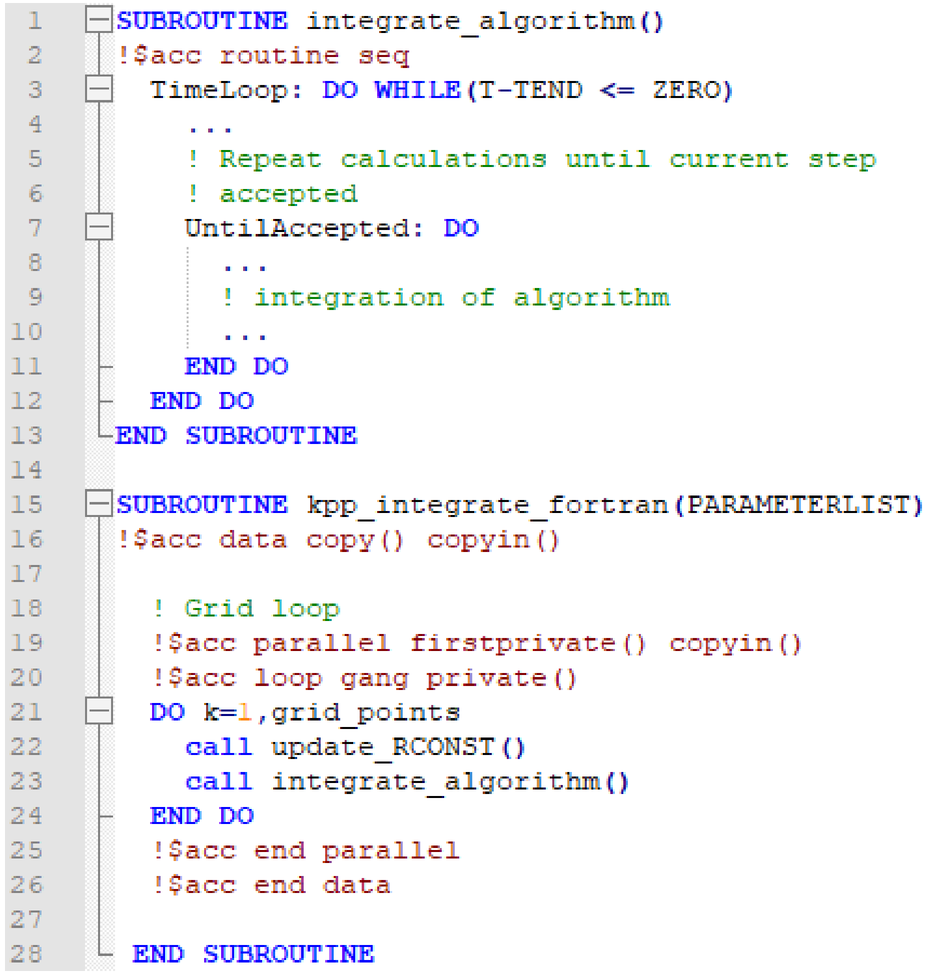
- try with gang and gang vector pragma

gang works fine in test setup

but: with 1 “thread” per SM GPU not fully used

gang vector crashes if > 1000 grid points

ongoing: fix issue with debugger



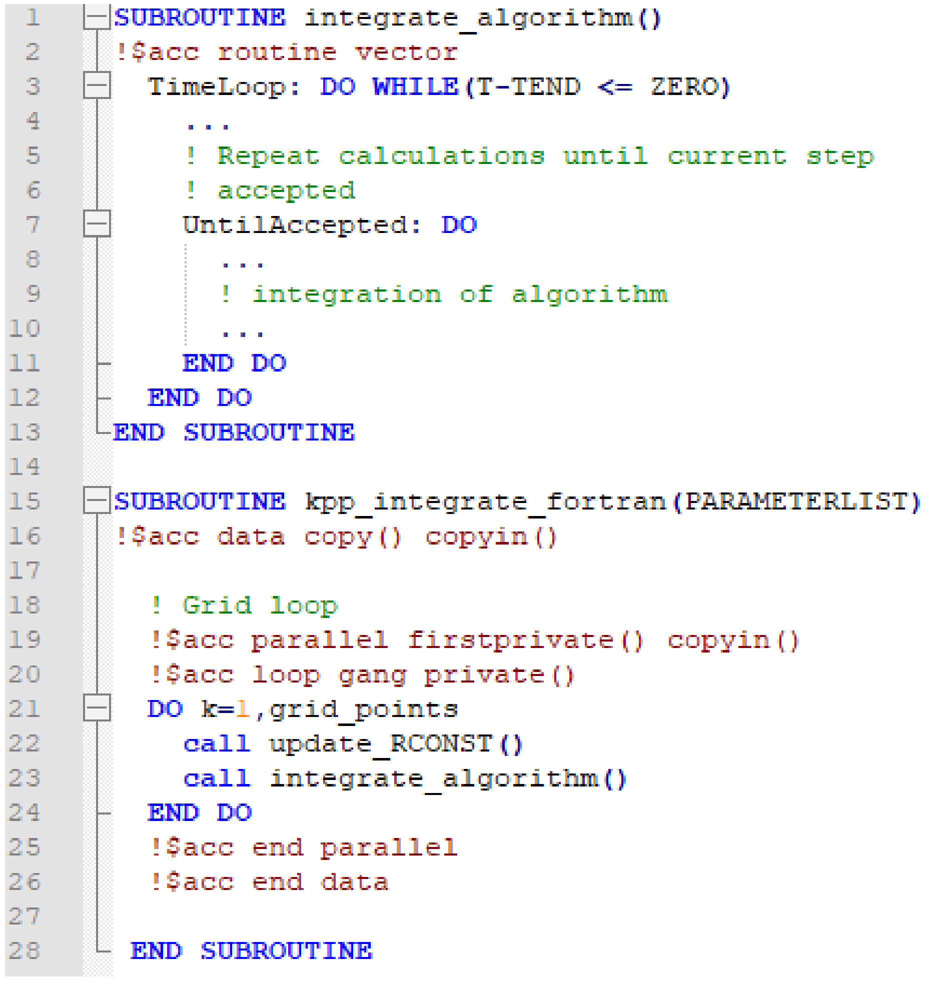
```
SUBROUTINE integrate_algorithm()
!$acc routine seq
  TimeLoop: DO WHILE(T-TEND <= ZERO)
    ...
    ! Repeat calculations until current step
    ! accepted
    UntilAccepted: DO
      ...
      ! integration of algorithm
      ...
    END DO
  END DO
END SUBROUTINE

SUBROUTINE kpp_integrate_fortran(PARAMETERLIST)
!$acc data copy() copyin()

  ! Grid loop
  !$acc parallel firstprivate() copyin()
  !$acc loop gang private()
  DO k=1,grid_points
    call update_RCONST()
    call integrate_algorithm()
  END DO
  !$acc end parallel
  !$acc end data

 END SUBROUTINE
```

OpenACC IN MECCA KPP

Implementation - Second variant

- **Second variant:** make use of working gang loop
- Try to accelerate the inner routines with acc routine vector statements

doesn’t work for the integration algorithm because compiler tries to parallelize the time loop



```
SUBROUTINE integrate_algorithm()
!$acc routine vector
  TimeLoop: DO WHILE(T-TEND <= ZERO)
    ...
    ! Repeat calculations until current step
    ! accepted
    UntilAccepted: DO
      ...
      ! integration of algorithm
      ...
    END DO
  END DO
END SUBROUTINE

SUBROUTINE kpp_integrate_fortran(PARAMETERLIST)
!$acc data copy() copyin()

  ! Grid loop
  !$acc parallel firstprivate() copyin()
  !$acc loop gang private()
  DO k=1,grid_points
    call update_RCONST()
    call integrate_algorithm()
  END DO
  !$acc end parallel
  !$acc end data

 END SUBROUTINE
```

OpenACC FOR MECCA KPP

Current work and plans

Further alternatives based on an approach to parallelize the calculations inside the loops of the integration algorithm are currently tested:

- using cublas library for calculating the LU decomposition and solving the ODE
  - ➢works, but large overhead relative to few calculations per kernel
- moving a part of the outer grid loop to the most inner loop (grid block loop instead of individual grid points) by using the `vectormode` of kp4 from KPP
  - ➢more calculations can be parallelized and calls to cublas library can be made in a batch-like approach
  - ➢the `vectormode` test code with OpenACC is work-in-progress

OpenACC IN JVAL

Background

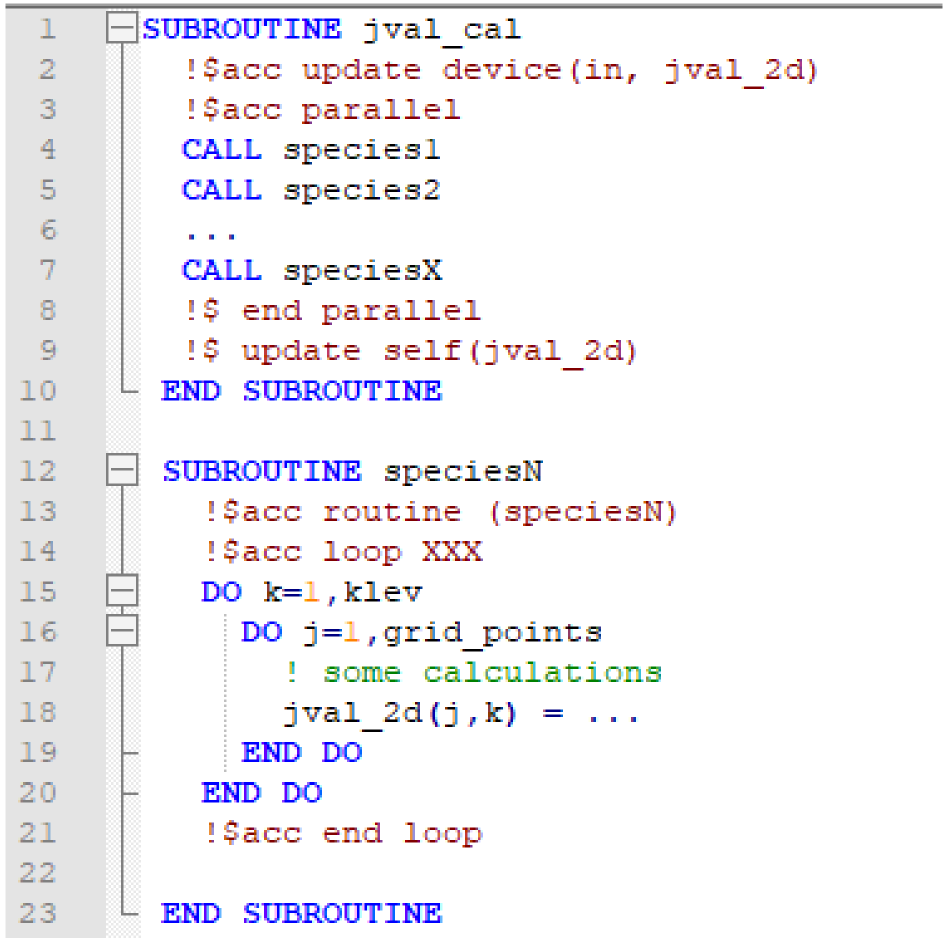
JVAL	Calculates photolysis rates for species of a chemical mechanism: based on lookup tables and polynomial fits
	not next most expensive routine (even excluding those benefiting from KPP speedup) but partly based on automatic code generation, thus easier to test different approaches

Relatively most expensive part (around 75 %) of JVAL is the calculation of photolysis rates, based on input fields like O2, O3 and temperature

OpenACC IN JVAL

Implementation

- one combined kernel makes up about 25 % of GPU time for about 100 chemical species
- the other 75 % are due to data transfers, adding latency
- `jval_cal` a bit less than 2x more expensive than on CPU, kernel computations themselves much faster than original computation



```
SUBROUTINE jval_cal
  !$acc update device(in, jval_2d)
  !$acc parallel
  CALL species1
  CALL species2
  ...
  CALL speciesX
  !$ end parallel
  !$ update self(jval_2d)
 END SUBROUTINE

 SUBROUTINE speciesN
   !$acc routine (speciesN)
   !$acc loop XXX
   DO k=1,klev
     DO j=1,grid_points
       ! some calculations
       jval_2d(j,k) = ...
     END DO
   END DO
   !$acc end loop

 END SUBROUTINE
```

SUMMARY AND OUTLOOK

Working CUDA implementation for MECCA and SCAV modules with a good speedup on JUWELS booster. OpenACC developments are ongoing.

Next:

− potentially adapt CUDA KPP parser Medina for GMXe

− port MESSy submodels using OpenACC

  - find versatile OpenACC implementation for KPP of similar speed as CUDA approach
  - use ported KPP in SCAV and GMXe
  - port non-KPP parts of SCAV and further submodules

– develop mechanism in interface layer of MESSy to share data on GPU across submodules (and initiate data transfers if required), e.g., based on managed memory

Thank you for your attention!

Questions?

LITERATURE

1: Jöckel, P et al.: Earth System Chemistry integrated Modelling (ESCiMo) with the Modular Earth Submodel System (MESSy) version 2.51, GMD, 9, 1153–1200, doi: 10.5194/gmd-9-1153-2016, 2016

2: Chowdhury, S. et al.: Global and national assessment of the incidence of asthma in children and adolescents from major sources of ambient NO2, Env. Research Letters, doi:10.1088/1748-9326/abe909, 2021.

3: Pozzer, A. et al.: Regional and global contributions of air pollution to risk of death from COVID-19, Cardiovascular Research, doi:10.1093/cvr/cvaa288, 2020

4: Johansson, S. et al..: Pollution trace gas distributions and their transport in the Asian monsoon upper troposphere and lowermost stratosphere during the StratoClim campaign 2017, ACP, 20, 14 695–14 715, doi:10.5194/acp-20-14695-2020, 2020

5: Bourtsoukidis, E. et al.: Non-methane hydrocarbon (C2–C8) sources and sinks around the Arabian Peninsula, ACP, 19, 7209–7232, doi: 10.5194/acp-19-7209-2019, 2019

6: Sander, R et al.: The community atmospheric chemistry box model CAABA/MECCA-4.0, GMD,12, 1365–1385, 12-1365-2019, 2019.

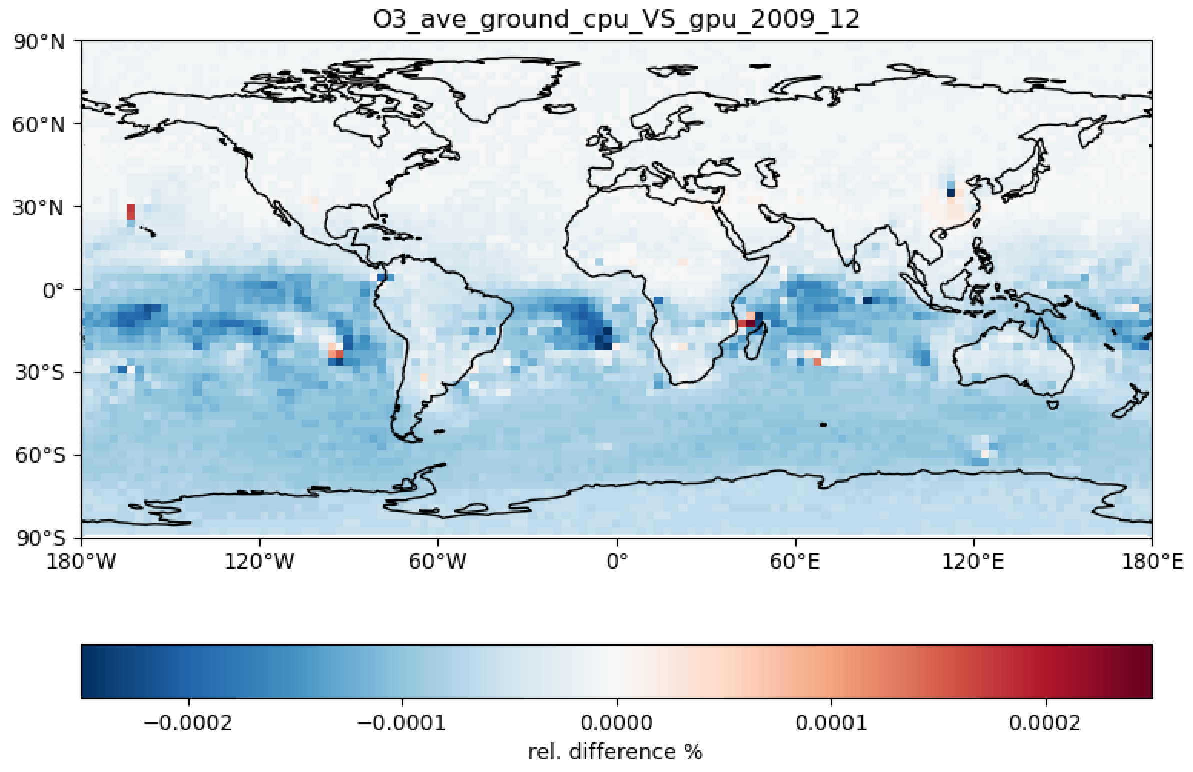
7: V. Damian, et al.: ``The Kinetic PreProcessor KPP — A Software Environment for Solving Chemical Kinetics'', Computers and Chemical Engineering, Vol. 26, No. 11, p. 1567-1579, 2002.

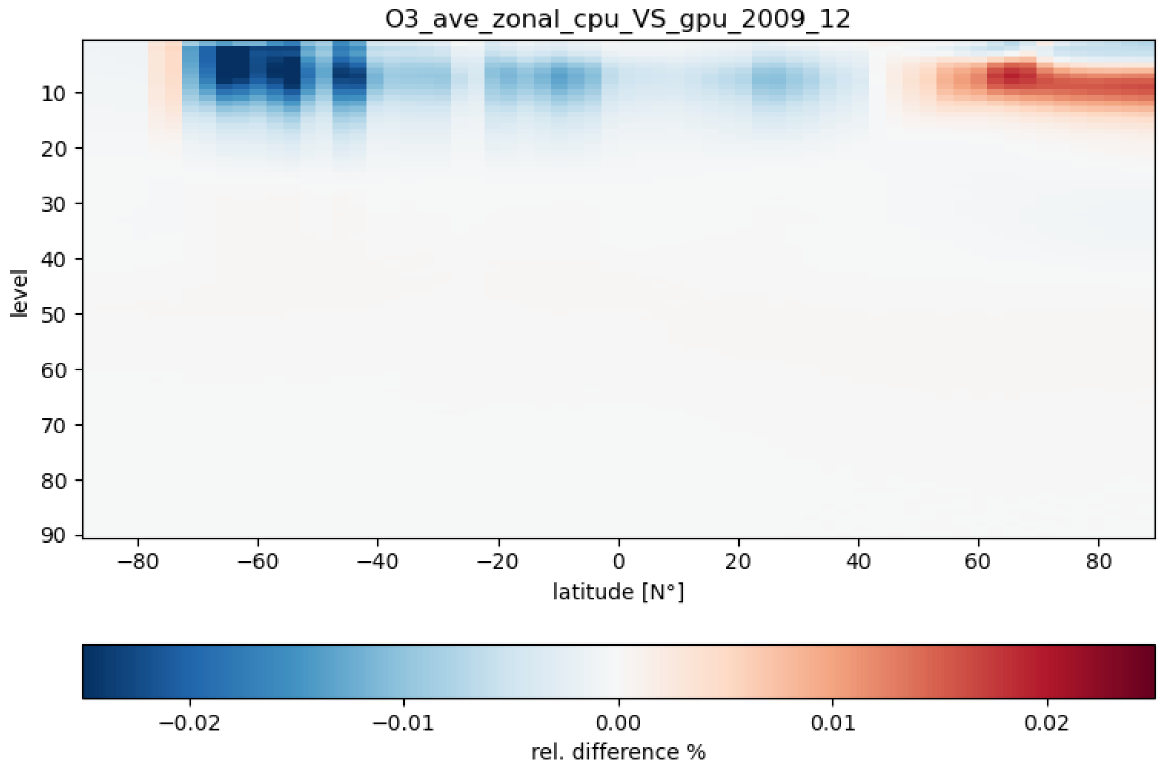
8: Alvanos, M. and Christoudias, T., 2017. MEDINA: MECCA Development in Accelerators – KPP Fortran to CUDA source-to-source Pre-processor. Journal of Open Research Software, 5(1)

9: Sander, R., et al..: The photolysis module JVAL-14, compatible with the MESSy standard, and the JVal PreProcessor (JVPP), GMD, 7, 2653–2662, 7-2653-2014, 2014.

MEDINA EVALUATION

Stability in long-term simulation, CCMI2 mechanism


O3_ave_ground_cpu_VS_gpu_2009_12
90°N
60°N
30°N
0°
30°S
60°S
90°S
180°W
120°W
60°W
0°
60°E
120°E
180°E
−0.0002
−0.0001
0.0000
0.0001
0.0002
rel. difference %


O3_ave_zonal_cpu_VS_gpu_2009_12
level
latitude [N°]
rel. difference %