Sprint 5

Expanding the MESSy infrastructure for CPU/GPU memory management

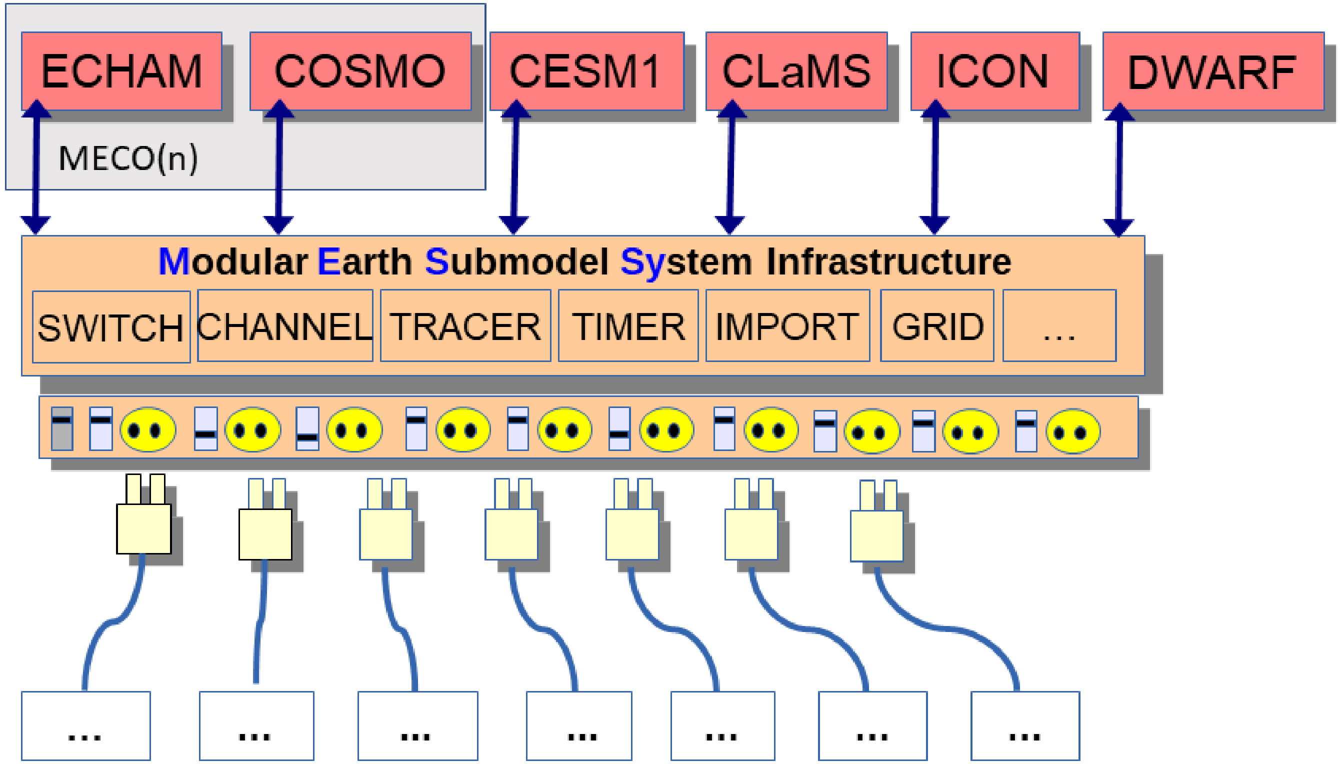
Astrid Kerkweg (FZ Jülich), Patrick Jöckel (DLR), Enrico Degregori (DKRZ), Timo Kirfel (FZ Jülich), Kerstin Hartung (DLR)

General information

- MESSy is a software project …
  - ➢ … providing an infrastructure for coupling atmospheric legacy models (ICON, ECHAM, COSMO) to specialised ESM components, the so-called submodels
  - ➢ … providing a collection of specialised ESM components (submodels) like
    - ➢ … chemistry packages (as kinetic solvers, dry deposition and scavenging of tracer gases etc.
    - ➢ … physical parametrisations (e.g., cloud, convection, …)
    - ➢ … diagnostics (e.g. output on tracks of measurement platforms as aircraft, balloons… , calculation of iso-surfaces, …)
- MESSy is used e.g., …
  - (predominantly) as chemistry-climate model (CCM, e.g. in CMIP6 or CCMI, Jöckel et al., 2016)
  - for process understanding, e.g. in detailed studies of multiphase chemistry (e.g., Franco et al., 2021, Rosanka et al., 2023)
  - in idealised studies (Garny et al., 2020)
  - …

General information

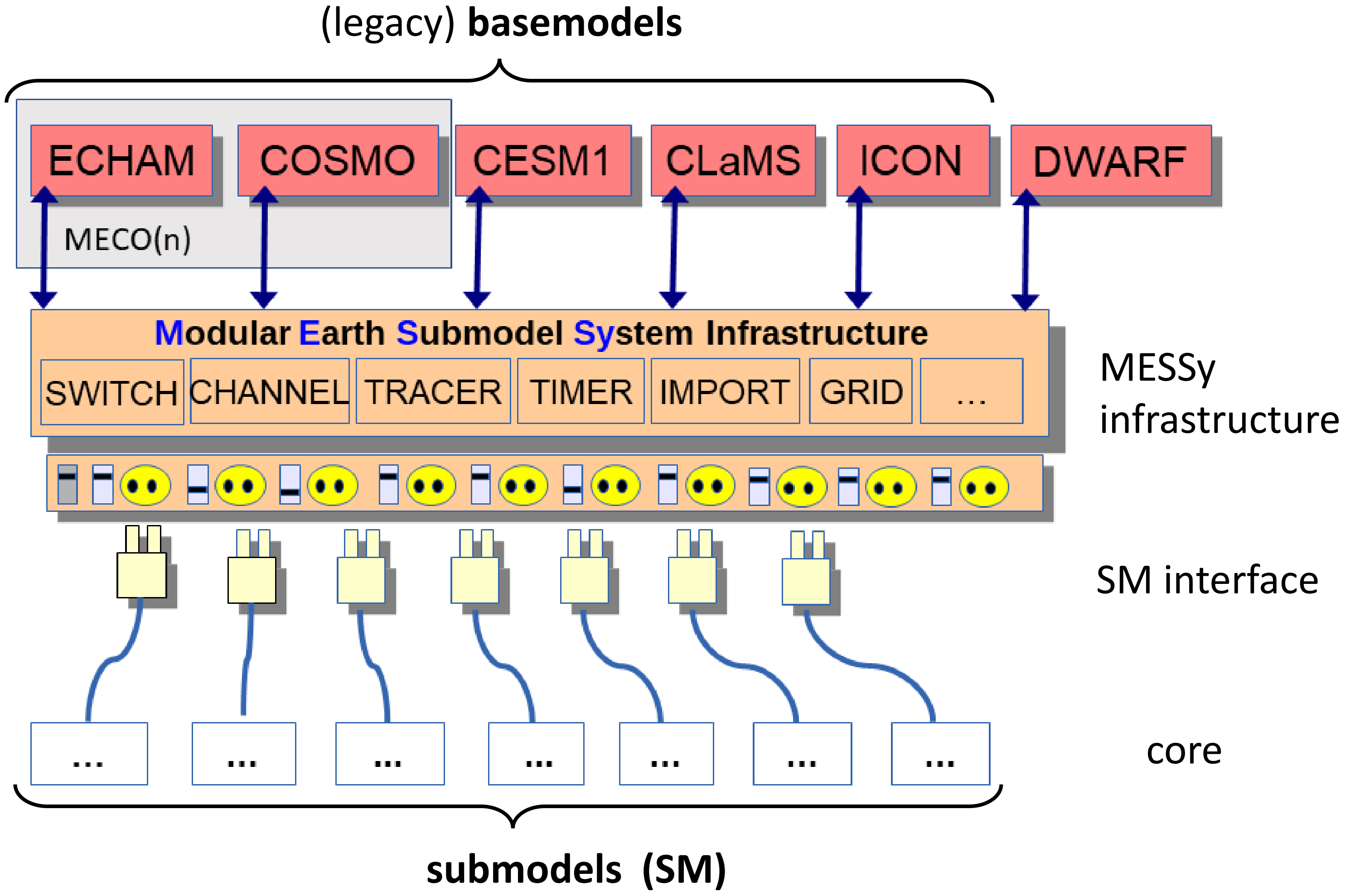
- MESSy contains a large code base
  - ➢ code can not be ported to GPU at once



courtesy of P. Jöckel (DLR), modif. by A. Kerkweg (FZJ)

General information

- MESSy contains a large code base
  - ➢ code can not be ported to GPU at once

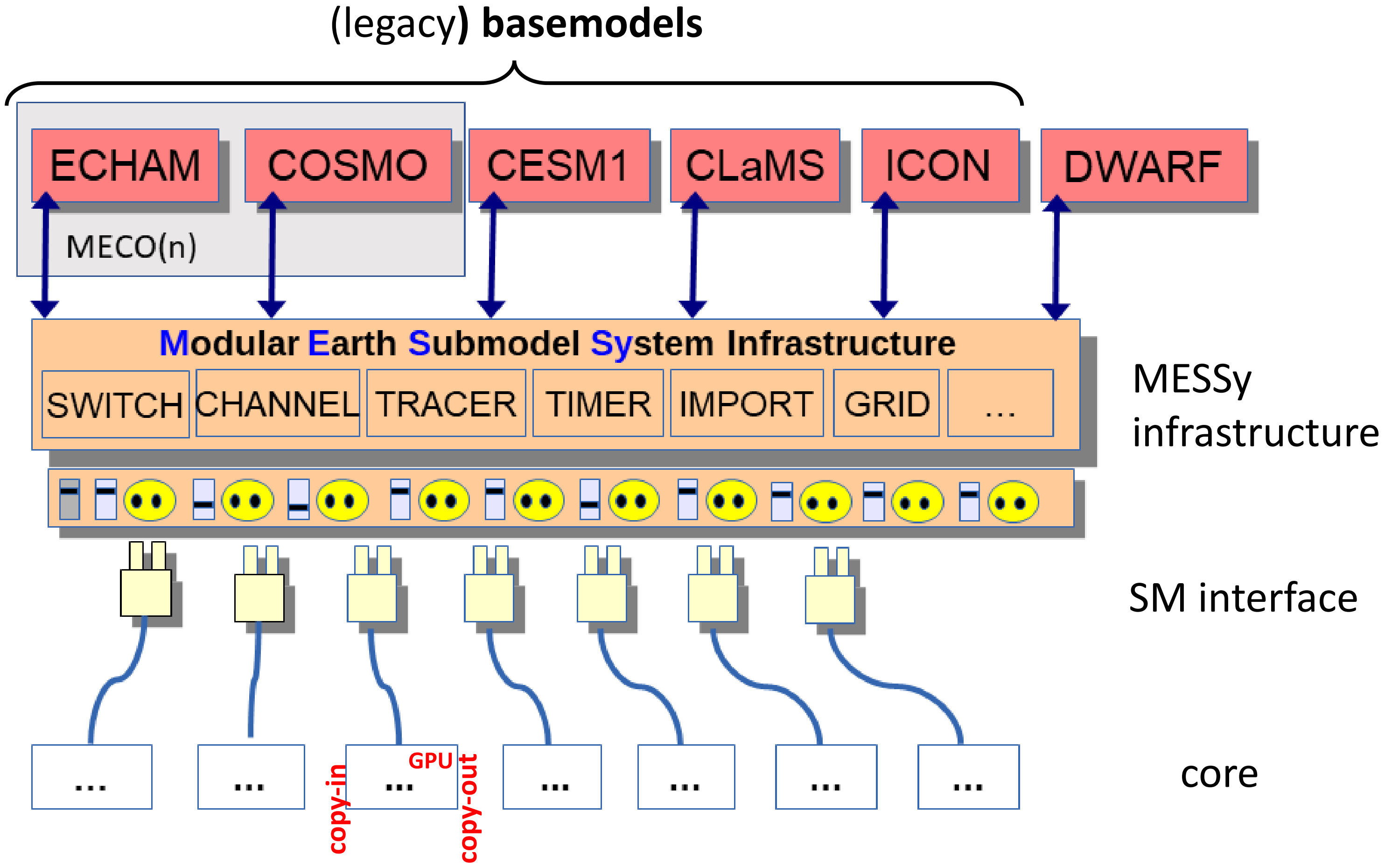


courtesy of P. Jöckel (DLR), modif. by A. Kerkweg (FZJ)

Forschungszentrum JÜLICH
DLR Deutsches Zentrum für Luft- und Raumfahrt German Aerospace Center

General information

- MESSy contains a large code base
  - ➢ code can not be ported to GPU at once
  - ➢ porting individual processes leads to large overhead due to too many data copies



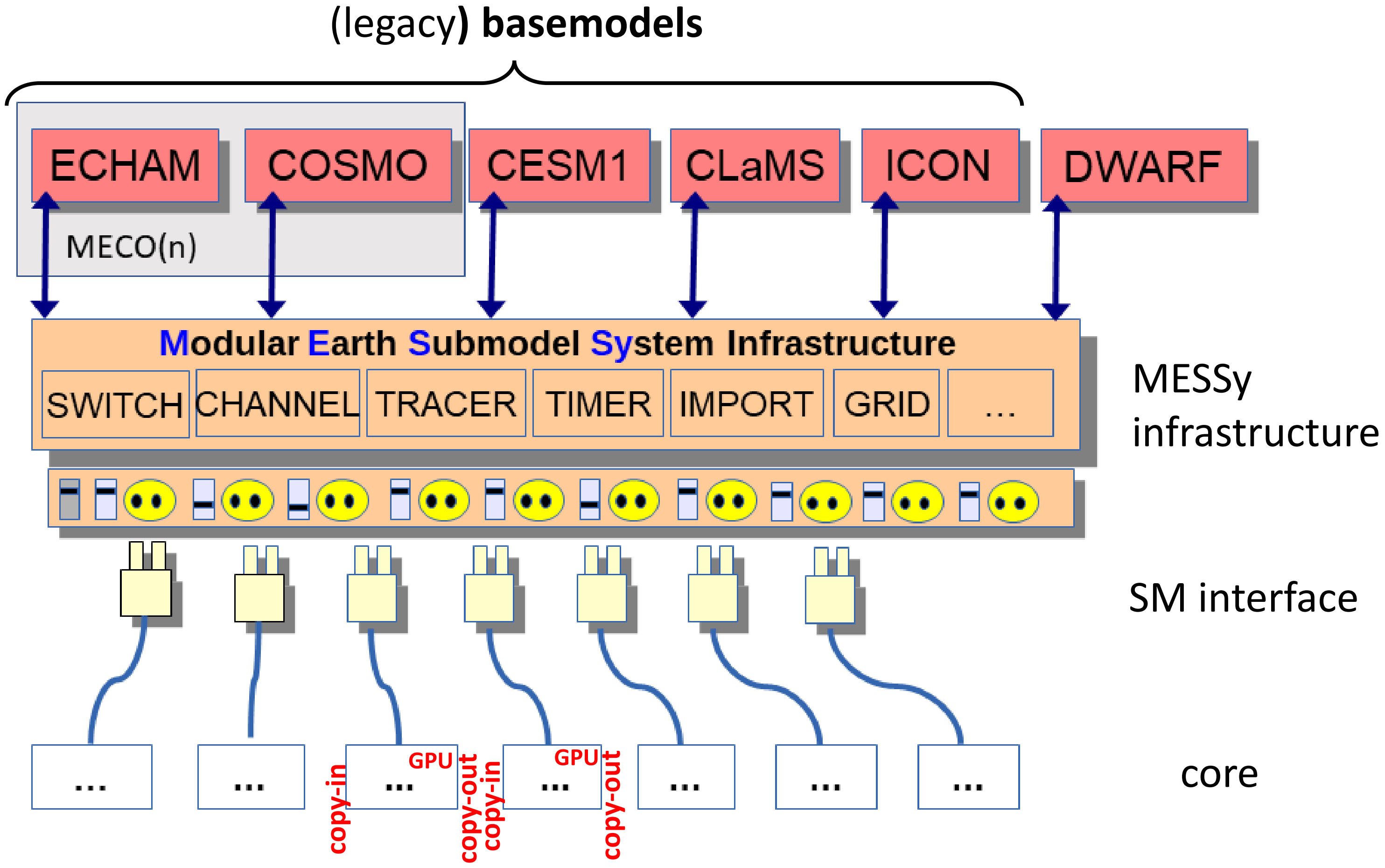
courtesy of P. Jöckel (DLR), modif. by A. Kerkweg (FZJ)

JÜLICH Forschungszentrum

DLR Deutsches Zentrum für Luft- und Raumfahrt German Aerospace Center

General information

- MESSy contains a large code base
  - ➢ code can not be ported to GPU at once
  - ➢ porting individual processes leads to large overhead due to too many data copies
  - ➢ infrastructure expansion for efficient copy-strategy required



courtesy of P. Jöckel (DLR), modif. by A. Kerkweg (FZJ)

JÜLICH Forschungszentrum

DLR Deutsches Zentrum für Luft- und Raumfahrt German Aerospace Center

General information

Sprint task: develop a concept for a MESSy infrastructure expansion allowing for an efficient data transfer between host and device

Sprint duration: 4 month

natESM programmer: Enrico Degregori

Results: clarification of terms

the MESSy infrastructure submodel CHANNEL provides an interface for the flexible and efficient data exchange / sharing between different processes (submodels).

- channel objects, representing data fields including their meta information (e.g. attributes) and their underlying geometric structure (representation),
- channels, representing sets of “related” channel objects with additional meta information. The “relation” can be, for instance, the simple fact that the channel objects are defined by the same submodel.

Functions:

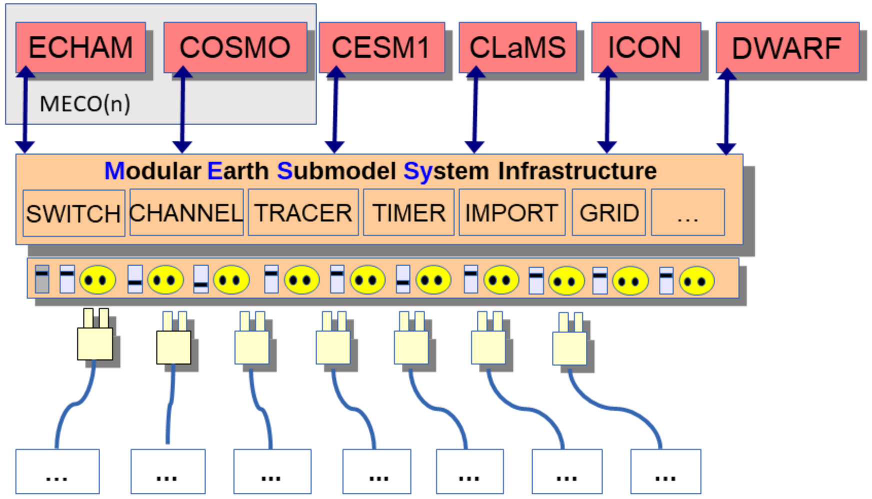
CALL new_channel_objects => creates new data object

CALL get_channel_object => set pointer to data field

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

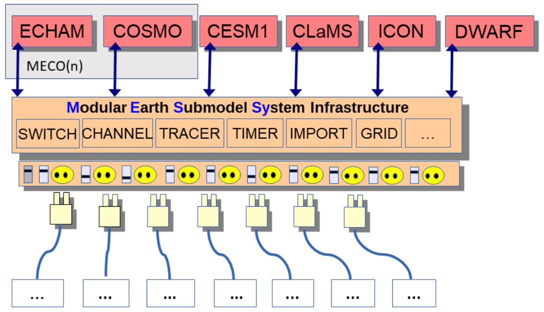
Challenges

- The big MESSy code base (internal coupling to 5 legacy basemodels) => long familiarisation period required



courtesy of P. Jöckel (DLR), modif. by A. Kerkweg (FZJ)

Challenges



courtesy of P. Jöckel (DLR), modif. by A. Kerkweg (FZJ)

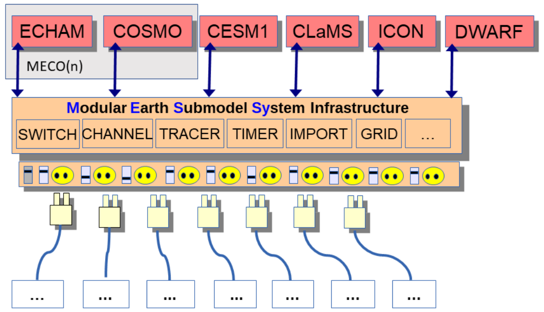
- The big MESSy code base (internal coupling to 5 legacy basemodels) => long familiarisation period required
- Solution: use MESSy DWARF
  - simplified configuration, no legacy basemodel: DWARF basemodel consists of, an initialisation phase, time loop and finalisation phase

```
PROGRAM dwarf
[...]
IMPLICIT NONE

! ######### INITIALISATION PHASE ###################
CALL messy_setup(0)
[...]
CALL messy_initialize        ! read submodel namelists
CALL messy_new_tracer        ! define tracers
CALL messy_init_memory       ! allocated submodel memory
[...]
CALL messy_init_coupling     ! couple to other submodels
CALL messy_read_restart      ! read restart information for all channels
CALL messy_init_tracer       ! initilialise tracers

! ###### TIME LOOP / INTEGRATION PHASE ############
 time_loop: DO WHILE ( .NOT. lbreak)
    ! reset time
    CALL messy_time(1)
    CALL messy_tendency_reset
    CALL messy_global_start
    [...]
    CALL messy_physc
    [...]
    CALL messy_global_end
    ! write output
    CALL messy_write_output
    ! write restart file
    CALL messy_write_restart
    ! step to next time step / set new dates
    CALL messy_time(2)
END DO time_loop

! ###### FINALIZING PHASE / FREE MEMORY ##########
CALL messy_free_memory   ! close output file, deallocate fields
CALL messy_finalize      ! finalize MPI environment

END PROGRAM dwarf
```

JÜLICH Forschungszentrum

DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Challenges

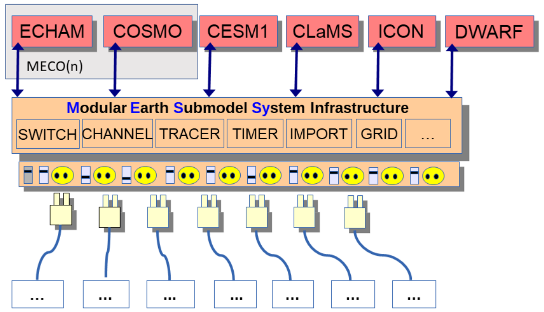


courtesy of P. Jöckel (DLR), modif. by A. Kerkweg (FZJ)

- The big MESSy code base (internal coupling to 5 legacy basemodels) => long familiarisation period required
- Solution: use MESSy DWARF
  - simplified configuration, no legacy basemodel: DWARF basemodel consists of, an initialisation phase, time loop and finalisation phase
  - add simplified MESSy submodels GPU1, GPU2, GPU3 using the most important infrastructure components for memory management (CHANNEL, TRACER and TENDENCY)

```
PROGRAM dwarf
[...]
CALL messy_init_memory  ! allocated submodel memory
[...]

! ###### TIME LOOP / INTEGRATION PHASE ############
 time_loop: DO WHILE ( .NOT. lbreak)
      [...]
      CALL messy_physc
      [...]
END DO time_loop

! ###### FINALIZING PHASE / FREE MEMORY ##########
[...]
END PROGRAM dwarf

=== SUBMODEL  CONTROL ===
[...]
SUBROUTINE messy_init_memory
 USE messy_main_switch      ! ONLY: USE_*
 USE messy_main_control     ! ONLY: entrypoint, subentry
 USE messy_main_channel_bi,  ONLY: main_channel_init_memory
 USE messy_gpu1_si,          ONLY: gpu1_init_memory
[...]
CALL channel_init_memory
CALL data_init_memory
[...]
IF (USE_GPU1) CALL gpu1_init_memory
IF (USE_GPU2) CALL gpu2_init_memory
IF (USE_GPU3) CALL gpu3_init_memory
[...]
END SUBROUTINE messy_init_memory
! ------------------------------------------------------------------------------------
[...]
! ------------------------------------------------------------------------------------
SUBROUTINE messy_physc

 USE messy_main_switch      ! ONLY: USE_*
 USE messy_main_control     ! ONLY: entrypoint, subentry
 USE messy_gpu1_si,          ONLY: gpu1_physc
 USE messy_gpu2_si,          ONLY: gpu2_physc
 USE messy_gpu3_si,          ONLY: gpu3_physc
[...]
IF (USE_GPU1) CALL gpu1_physc
IF (USE_GPU2) CALL gpu2_physc
IF (USE_GPU3) CALL gpu3_physc
[...]
END SUBROUTINE messy_physc
```

Challenges



courtesy of P. Jöckel (DLR), modif. by A. Kerkweg (FZJ)

- The big MESSy code base (internal coupling to 5 legacy basemodels) => long familiarisation period required
- Solution: use MESSy DWARF ExaESM
  - simplified configuration, no legacy basemodel:
    DWARF basemodel consists of, an initialisation phase, time loop and finalisation phase
  - add simplified MESSy submodels GPU1, GPU2, GPU3 using the most important infrastructure components for memory exchange (CHANNEL, TRACER and TENDENCY)

This DWARF was provided at the beginning of the sprint
=> concentration on basic infrastructure elements possible
=> no need of going into details and specifics of individual (legacy) basemodels

JÜLICH Forschungszentrum
DLR Deutsches Zentrum für Luft- und Raumfahrt German Aerospace Center

Results: overview

- Based on this DWARF setup Enrico developed a concept and already implemented it
- in addition to the GPU1-3 submodels, he tested MECCA (which core was already ported to GPUs via CUDA before).
- requirements for the applicability within a full MESSy setup were discussed (and implemented during and after the sprint – mainly by Astrid Kerkweg)
- additional developments required for the application when MESSy is coupled to a full dynamical model were discussed (see outlook).

Results: the concept

logging of GPU related information takes place via the channel object meta-data

- lopenacc – is a channel object at all required on GPU?
- location – on which device resides the most updated data?
  (MEMORY_HOST, MEMORY_HOST_DEVICE, MEMORY_DEVICE)
- mem_id / oriobj – special treatment of channel objects which memory was allocated somewhere else
  (added after the sprint – examples were not included in DWARF test setup)

Deutsches Zentrum für Luft- und Raumfahrt German Aerospace Center

Results: the concept

logging of GPU related information takes place via the channel object meta-data

- lopenacc – is a channel object at all required on GPU?
- location – on which device resides the most updated data?
  (MEMORY_HOST, MEMORY_HOST_DEVICE, MEMORY_DEVICE)
- mem_id / oriobj – special treatment of channel objects which memory was allocated somewhere else
  (added after the sprint – examples were not included in DWARF test setup)

=> memory access MUST always proceed via

- get_channel_object calls or
- (for prognostic variables) via TENDENCY

JÜLICH Forschungszentrum
DLR Deutsches Zentrum für Luft- und Raumfahrt German Aerospace Center

Results: the concept

logging of GPU related information takes place via the channel object meta-data

- lopenacc – is a channel object at all required on GPU?
- location – on which device resides the most updated data?
  (MEMORY_HOST, MEMORY_HOST_DEVICE, MEMORY_DEVICE)
- mem_id / oriobj – special treatment of channel objects which memory was allocated somewhere else
  (added after the sprint – examples were not included in DWARF test setup)

=> memory access MUST always proceed via

- get_channel_object calls or
- (for prognostic variables) via TENDENCY

- check / copies always happens at the beginning of a submodel call
- GPU infrastructure expansion is fully hidden behind usual API calls

=> when coding a submodel as usual for CPU the developer does not need to explicitly take care of the GPU expansion

Results: consequences

- the MESSy infrastructure and process submodels **must only use** the API routines of CHANNEL and TENDENCY to access memory
- get_channel_object calls are required **every time step** to check the memory location and trigger copy to/from device (consistent with ICON implementation)
- prognostic variables can only be accessed / modified using TENDENCY routines
- tracers need to be re-ordered into 2 blocks (one only CPU, one on GPU+ CPU) to optimise the memory allocation on the device

Results: consequences

- the MESSy infrastructure and process submodels **must only use** the API routines of CHANNEL and TENDENCY to access memory
- get_channel_object calls are required **every time step** to check the memory location and trigger copy to/from device (consistent with ICON implementation)
- prognostic variables can only be accessed / modified using TENDENCY routines
- tracers need to be re-ordered into 2 blocks (one only CPU, one on GPU+ CPU) to optimise the memory allocation on the device

=> cleanup of MESSy code required – concepts that previously existed must now be followed exactly and without exception
=> done during and after the sprint (mainly by Astrid Kerkweg)
=> status "acc_aware" created, i.e., a submodel triggers the memory update but is itself not ported to GPU. Submodels for CCMI2/RD1 setup have been made "acc_aware".

Results: Conclusion

- we are about to merge the GPU development branch into the MESSy development branch
- thanks to Enricos work, we not only have a concept, but on the MESSy side a fully implemented GPU infrastructure

Results: Conclusion

- we are about to merge the GPU development branch into the MESSy development branch
- thanks to Enricos work, we not only have a concept, but **on the MESSy side** a fully implemented GPU infrastructure

on the MESSy side ???

The legacy models do not use the MESSy infrastructure routines

=> procedure required on the MESSy side which triggers the basemodel data to be updated on the device the basemodel is running on.

Outlook & open questions

- **ESiWACE3** (Centre of Excellence in Simulation of Weather and Climate in Europe) successful application for support getting ICON/MESSy including the GPU infrastructure into production.
  - write routine to trigger data copies at the end of each entry point to the device ICON is running on
  - port MESSy infrastructure submodels where required to GPU
  - (optional) port further MESSy process submodels to GPU
  - benchmarking of GPU infrastructure expansion

JÜLICH Forschungszentrum
DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Outlook & open questions

- How is the **pay-off** between additional instructions to be followed to check if copies are required in contrast to the time saved by saving unnecessary copies between host and device?
- What is the consequence of **new chip developments** providing improved memory access times (between host and device)?
- Very much wanted (but most likely not to be found): a (always valid) **recipe** how to configure a simulation (GPU/CPU) dependent on the setup.

We need a **"real" case** (not DWARF, but ICON/MESSy or EMAC) to check the full consequences

JÜLICH Forschungszentrum

DLR Deutsches Zentrum für Luft- und Raumfahrt German Aerospace Center

Outlook & open questions

- How is the pay-off between additional instructio[illegible]owed to check if copies are required in contrast to the time s[illegible]ing unnecessary copies between host and device?
- What is the consequence of new[illegible]opments providing improved memory access times (between host[illegible])?
- Very much wanted (bu[illegible]y not to be found): a (always valid) recipe how to configure a sim[illegible]U/CPU) dependent on the setup.

We need a "r[illegible]ot DWARF, but ICON/MESSy or EMAC) to check the full consequence[illegible]

Thanks for your attention!