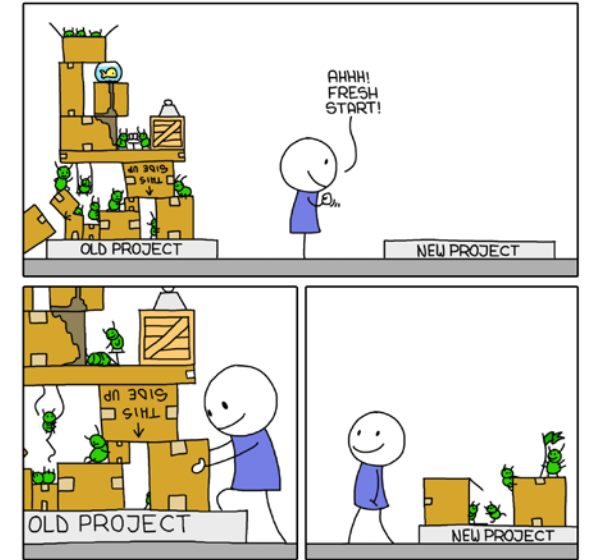


# Harmonized Emissions Component (HEMCO) Restructuring and interfacing with CAM

Haipeng Lin, Spring 2020

CODE REUSE

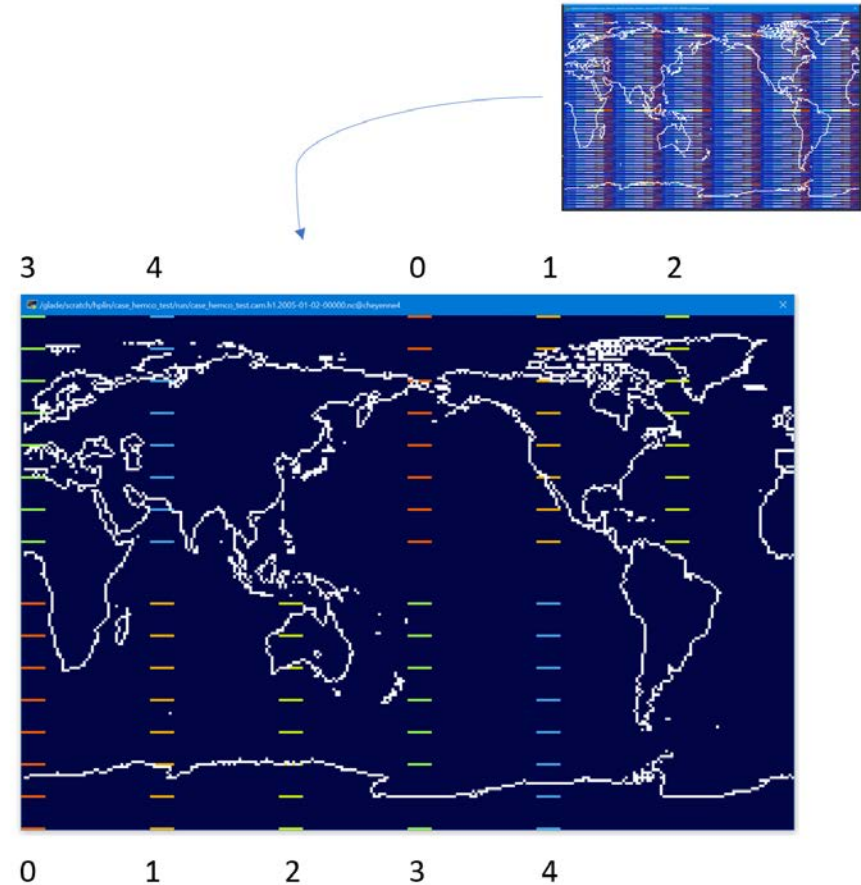
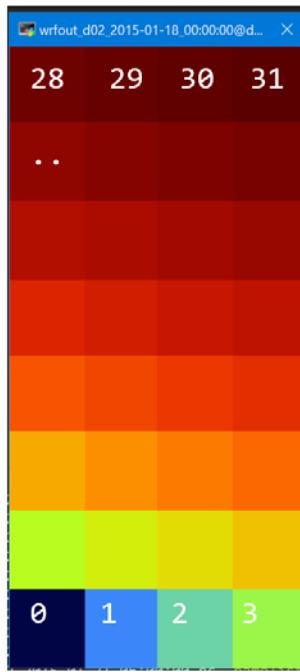


MONKEYUSER.COM

# Tale of technical constraints

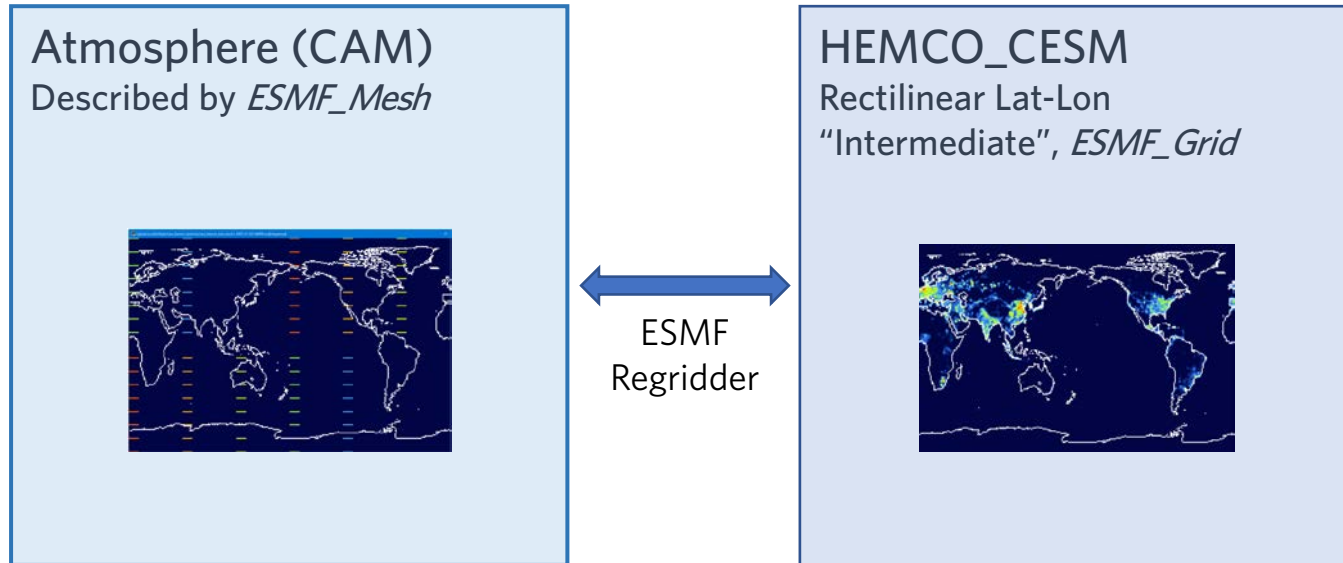
So I was explaining my proposed domain decomposition...

“No, no, no. You live in that world. That is 10 years behind. The problem is, we’re in this world.”



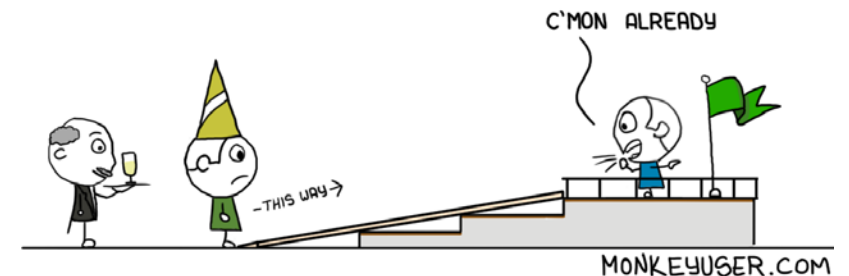
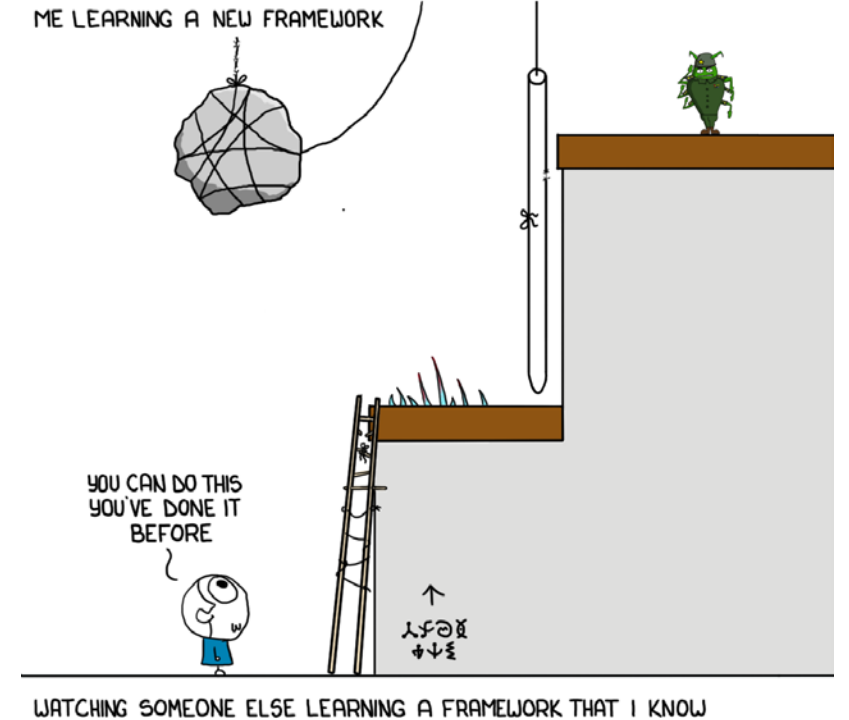
# CAM | ESMF | HEMCO "Hybrid" Architecture

- Must introduce an ESMF-based regrid layer



- Regridder requires running in a Gridded Component
- But the model must not be driven by ESMF!

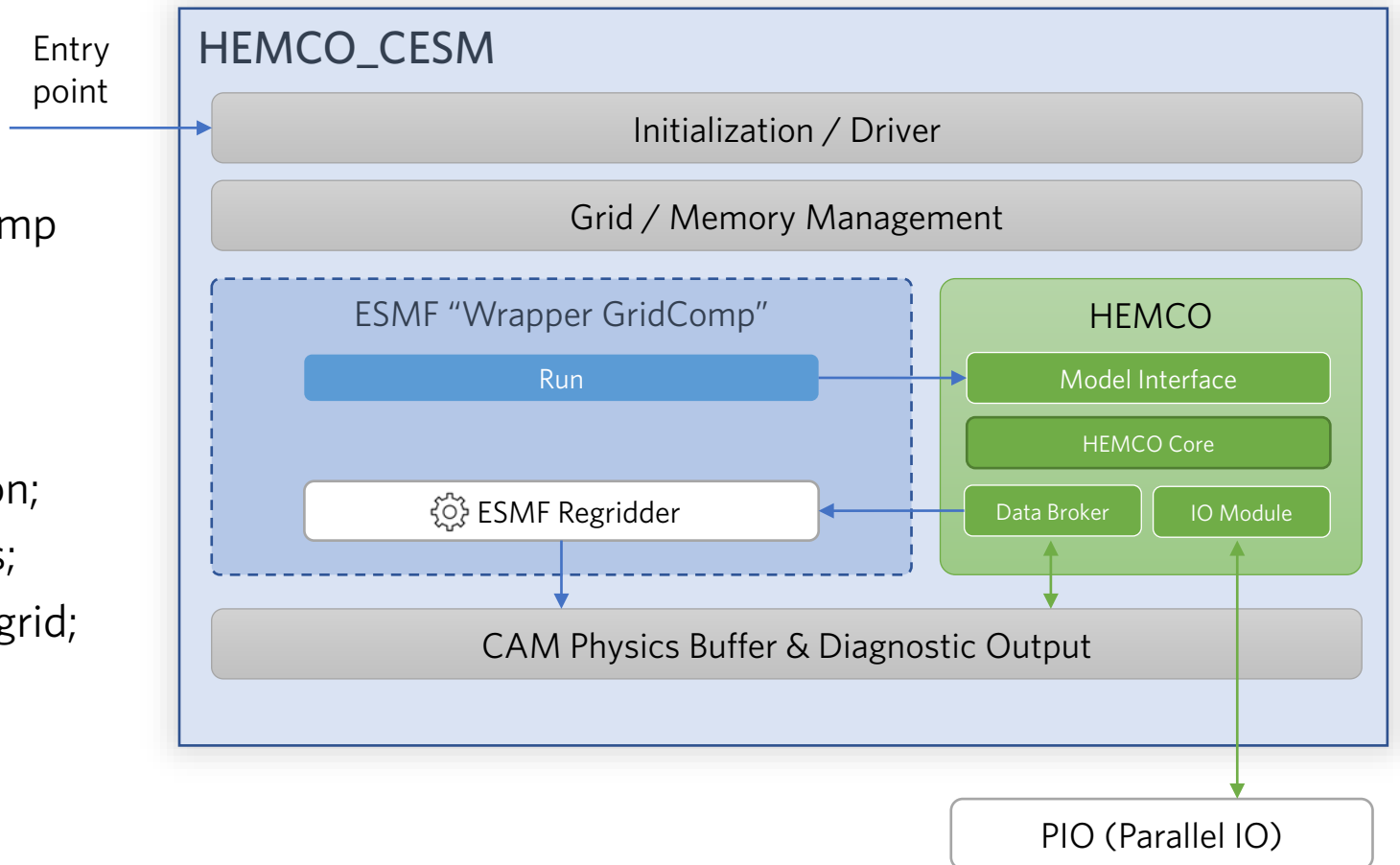
## DIFFERENT PERSPECTIVES



# Building a new “HEMCO standalone”

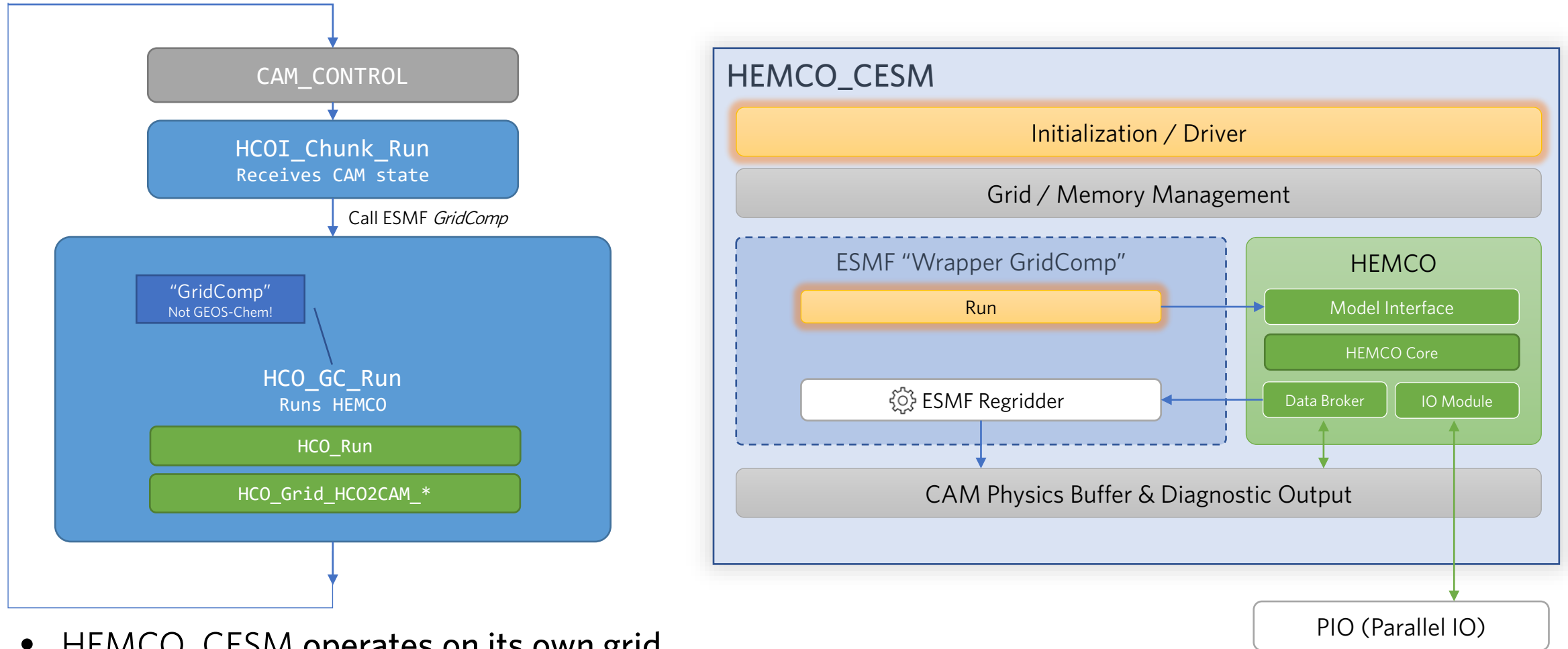
Main components:

- HEMCO Driver
  - “hemco\_interface.F90”
    - Creates HEMCO ESMF GridComp
    - “Controls ESMF”
- Grid Manager
  - “hco\_esmf\_grid.F90”
    - Copies “gc\_grid\_mod” for lat-lon;
    - Manages ESMF Regrid Handles;
    - Performs HEMCO <-> CAM regrid;
- Export Manager
  - “hco\_cam\_exports.F90”
    - Exports to CAM History
    - Exports to CAM Physics Buffer



# Building a new “HEMCO standalone”

What does building a model-in-a-model *really* look like?



- HEMCO\_CESM operates on its own grid.
  - Thus, HEMCO\_CESM must handle its own domain decomposition within MPI.
- HEMCO\_CESM is not *driven* by ESMF but *drives* ESMF!

*I promise I did not invent this...*

This implementation proudly brought to you by the CAM *WACCMX* interface

```
edyn_esmf.F90      68
edyn_geogrid.F90   69
edyn_grid_comp.F90 195
edyn_init.F90      196
edyn_maqgrid.F90   197
edyn_mpi.F90       198
edyn_mud.F90       240
edyn_mudcom.F90    241
edyn_mudmod.F90    242
edyn_muh2ccr.F90   243
edyn_params.F90    259
edyn_solve.F90     260
edynamo.F90        261
filter.F90         262
getapex.F90        263
heelis.F90         283
ionosphere_interface.F90 328
litr_module.F90    329
oplus.F90          330
roqgridder.F90     331
rqrd_mod.F90       366
savefield_waccm.F90 367
utils_mod.F90      368
wei05cc.F90        369

68
69
195
196
197
198
240
241
242
243
259
260
261
262
263
283
284
285
286
328
329
330
331
366
367
368
369
410
447
448
449
450
460
461

subroutine edyn_gcomp_init(comp, importState, exportState, clock, rc)
end subroutine edyn_gcomp_init

!-----
subroutine edyn_gcomp_run(comp, importState, exportState, clock, rc)
end subroutine edyn_gcomp_run
!-----
subroutine edyn_gcomp_final(comp, importState, exportState, clock, rc)
end subroutine edyn_gcomp_final

!-----
subroutine edyn_gcomp_SetServices(comp, rc)
end subroutine edyn_gcomp_SetServices

subroutine edyn_grid_comp_init(mpi_comm)
end subroutine edyn_grid_comp_init

subroutine edyn_grid_comp_run1(ionos_epotential_model_in, cols, cole, efx_phys, kev_phys, amie_in, ltr_in)
end subroutine edyn_grid_comp_run1

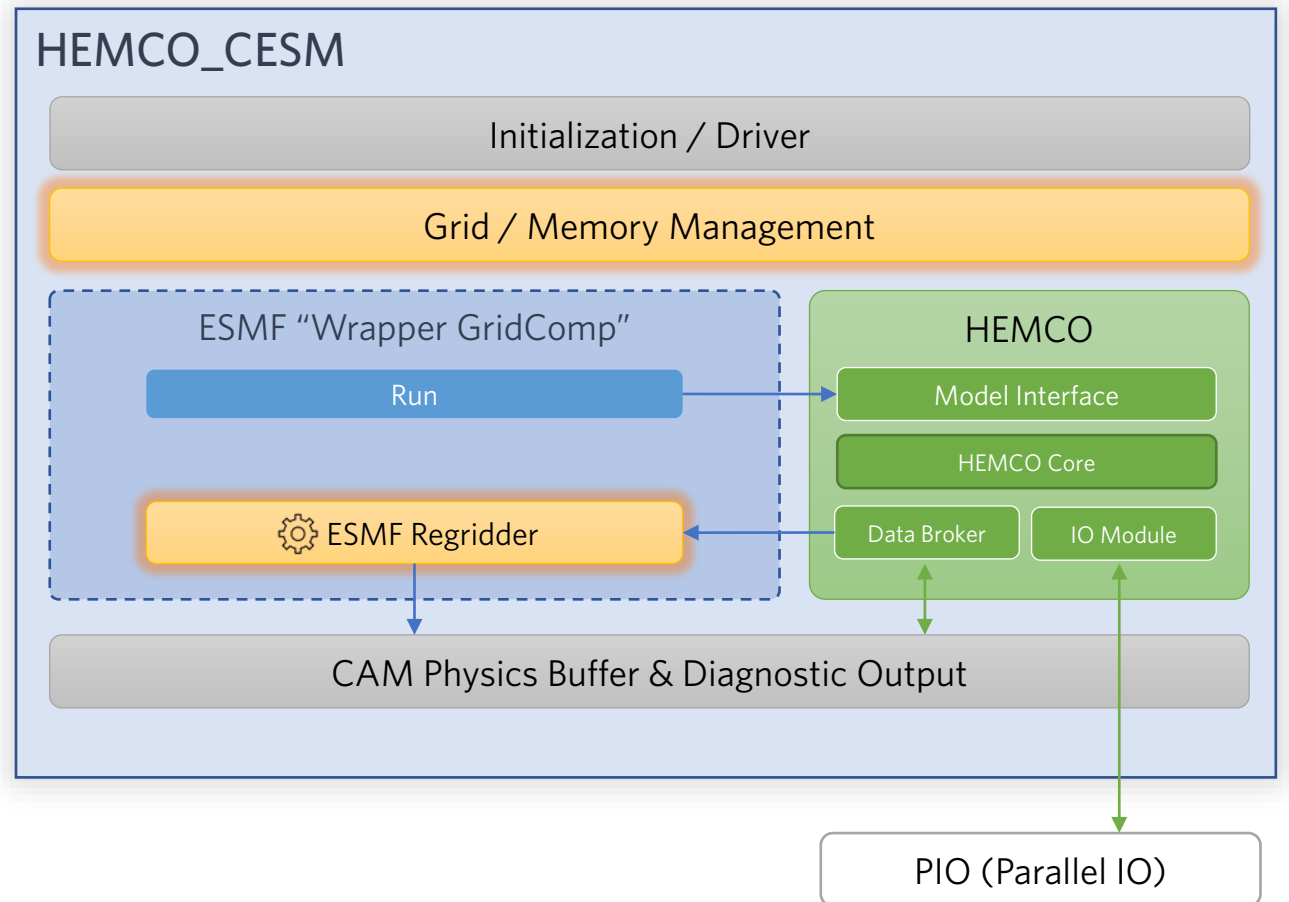
subroutine edyn_grid_comp_run2(omega_blk_in, pmid_blk_in, zi_blk_in, &
! Local variables
end subroutine edyn_grid_comp_run2

subroutine edyn_grid_comp_final()
end subroutine edyn_grid_comp_final
```

# Building a new “HEMCO standalone”

The ESMF regridder

```
42 !
43 !PUBLIC MEMBER FUNCTIONS:
44 !
45 public      :: HCO_Grid_Init
46 public      :: HCO_Grid_UpdateRegrid
47
48 public      :: HCO_Grid_HCO2CAM_2D      ! Regrid HEMCO to CAM mesh on 2D field
49 public      :: HCO_Grid_HCO2CAM_3D      ! ..on 3D field
50 public      :: HCO_Grid_CAM2HCO_2D      ! Regrid CAM mesh to HEMCO on 2D field
51 public      :: HCO_Grid_CAM2HCO_3D      ! ..on 3D field
52
53 !
54 !PRIVATE MEMBER FUNCTIONS:
55 !
56 private     :: HCO_Grid_ESMF_CreateCAM
57 private     :: HCO_Grid_ESMF_CreateCAMfield ! Create field on CAM physics mesh
58 private     :: HCO_Grid_ESMF_CreateHCO      ! Create HEMCO lat-lon grid in ESMF
59 private     :: HCO_Grid_ESMF_CreateHCOfield ! Create field on HEMCO ll grid
60
61 private     :: HCO_ESMF_Set2DHCO      ! Set ESMF field with 2D HEMCO data
62 private     :: HCO_ESMF_Set3DHCO      ! Set ESMF field with 3D HEMCO data
63 private     :: HCO_ESMF_Set2DCAM      ! Set ESMF field with 2D CAM mesh data
64 private     :: HCO_ESMF_Set3DCAM      ! Set ESMF field with 3D CAM mesh data
65
66 private     :: HCO_ESMF_Get1Dfield     ! Retrieve 1D ESMF field pointer
67 private     :: HCO_ESMF_Get2Dfield     ! Retrieve 2D ESMF field pointer
68 private     :: HCO_ESMF_Get3Dfield     ! Retrieve 3D ESMF field pointer
```



# Building a new “HEMCO standalone”

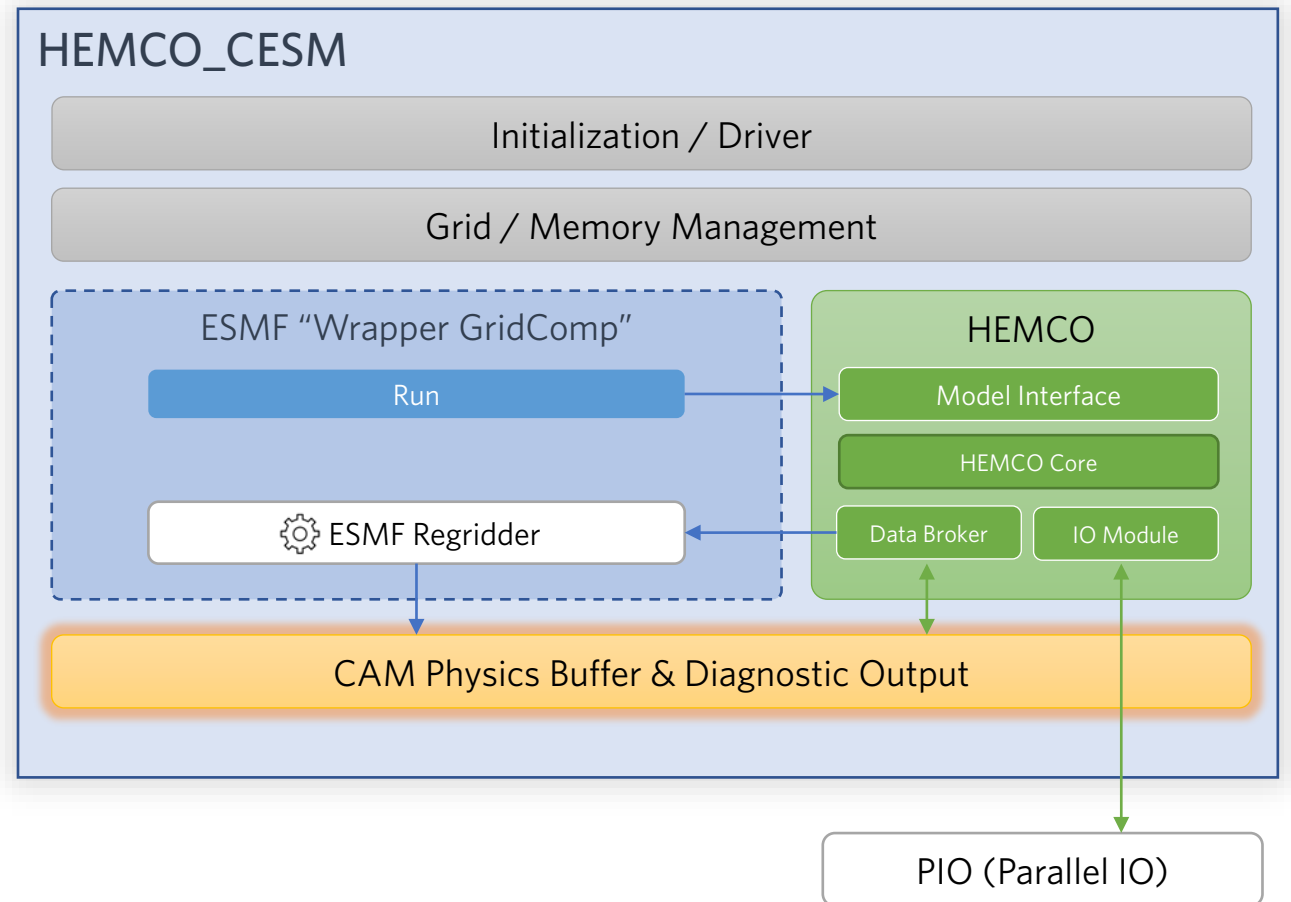
Interfacing with CAM from an exports component

```
48 !
49 ! !PUBLIC MEMBER FUNCTIONS:
50 !
51 public      :: HCO_Exports_Init
52 public      :: HCO_Export_History_HC03D
53 public      :: HCO_Export_History_CAM3D
54
55 public      :: HCO_Export_Pbuf_AddField
56 public      :: HCO_Export_Pbuf_CAM3D
```

HEMCO\_CESM is separate from all else!

- Runs in a “almost standalone” way (own grid, own memory)
- Writes emissions fluxes every timestep (30 minutes) to the “physics buffer”
- ... to be retrieved in whatever way the chemistry sees fit.

Bonus: Easy compatibility





# What does HEMCO need as input?

- Species Information

- ID, name, molecular weight, "emMW\_g", "MolecRatio", Henry's Law coef.
- From comments:

- Grid Information

- Meteorological input data

- Necessitates "State Conversion" from (whatever model) → "GEOS-FP"
- *Do we really need/want to use "GEOS-FP" for ExtState?*

```
! Emitted molecular weight of species [g/mol].
! Some hydrocarbon species (like ISOP) are emitted and
! transported as a number of equivalent carbon atoms.
! For these species, the emitted molecular weight will
! be 12.0 (the weight of 1 carbon atom).
HcoState%Spc(N)%EmMW_g      = SpcInfo%EmMW_g

! Emitted molecules per molecules of species [1].
! For most species, this will be 1.0. For hydrocarbon
! species (like ISOP) that are emitted and transported
! as equivalent carbon atoms, this will be the number
! of moles carbon per mole species.
HcoState%Spc(N)%MolecRatio = SpcInfo%MolecRatio
```

# Discussion: the “state conversion” and code-reuse issue

```
hcox_state_mod.F90
125 |-----|
126 | ! Met fields
127 | !-----|
128 |
129 TYPE(ExtDat_2R), POINTER :: U10M      ! E/W 10m wind speed [m/s]
130 TYPE(ExtDat_2R), POINTER :: V10M      ! N/S 10m wind speed [m/s]
131 TYPE(ExtDat_2R), POINTER :: ALBD      ! Surface albedo [-]
132 TYPE(ExtDat_2R), POINTER :: WLI      ! 0=water, 1=land, 2=ice
133 TYPE(ExtDat_2R), POINTER :: T2M      ! 2m Sfc temperature [K]
134 TYPE(ExtDat_2R), POINTER :: TSKIN    ! Surface skin temperature [K]
135 TYPE(ExtDat_2R), POINTER :: GWETROOT ! Root soil wetness [1]
136 TYPE(ExtDat_2R), POINTER :: GWETTOP  ! Top soil moisture [-]
137 TYPE(ExtDat_2R), POINTER :: SNOWHGT  ! Snow height [mm H2O = kg H2O/m2]
138 TYPE(ExtDat_2R), POINTER :: SNODP    ! Snow depth [m]
139 TYPE(ExtDat_2R), POINTER :: SNICE     ! Fraction of snow/ice [1]
140 TYPE(ExtDat_2R), POINTER :: USTAR    ! Friction velocity [m/s]
141 TYPE(ExtDat_2R), POINTER :: Z0       ! Sfc roughness height [m]
142 TYPE(ExtDat_2R), POINTER :: TROPP    ! Tropopause pressure [Pa]
143 TYPE(ExtDat_2R), POINTER :: SUNCOS   ! COS (SZA)
144 TYPE(ExtDat_2R), POINTER :: SZAFAC   ! current SZA/total daily SZA
145 TYPE(ExtDat_2R), POINTER :: PARDR    ! direct photosyn radiation [W/m2]
146 TYPE(ExtDat_2R), POINTER :: PARDF   ! diffuse photosyn radiation [W/m2]
147 TYPE(ExtDat_2R), POINTER :: PSC2_WET ! Interpolated sfc pressure [hPa]
148 TYPE(ExtDat_2R), POINTER :: RADSWG  ! surface radiation [W/m2]
149 TYPE(ExtDat_2R), POINTER :: FRCLND  ! Olson land fraction [-]
150 TYPE(ExtDat_2R), POINTER :: FRLAND  ! land fraction [-]
151 TYPE(ExtDat_2R), POINTER :: FROCEAN ! ocean fraction [-]
152 TYPE(ExtDat_2R), POINTER :: FRLAKE  ! lake fraction [-]
153 TYPE(ExtDat_2R), POINTER :: FRLANDIC ! land ice fraction [-]
154 TYPE(ExtDat_2R), POINTER :: CLDFRC  ! cloud fraction [-]
155 TYPE(ExtDat_2R), POINTER :: JNO2    ! J-Value for NO2 [1/s]
156 TYPE(ExtDat_2R), POINTER :: JOH     ! J-Value for O3->OH [1/s]
157 TYPE(ExtDat_2R), POINTER :: LAI     ! daily leaf area index [cm2/cm2]
158 TYPE(ExtDat_2R), POINTER :: CHLR    ! daily chlorophyll-a [mg/m3]
159 TYPE(ExtDat_2I), POINTER :: TropLev ! Tropopause level [1]
160 TYPE(ExtDat_2R), POINTER :: FLASH_DENS ! Lightning flash density [# /km2/s]
161 TYPE(ExtDat_2R), POINTER :: CONV_DEPTH ! Convective cloud depth [m]
162 INTEGER, POINTER :: PBL_MAX        ! Max height of PBL [level]
163 TYPE(ExtDat_3R), POINTER :: CNV_MFC ! Convective cloud mass flux [kg/m2/s]
164 TYPE(ExtDat_3R), POINTER :: FRAC_OF_PBL ! Fraction of grid box in PBL
165 TYPE(ExtDat_3R), POINTER :: SPHU    ! Spec. humidity [kg H2O/kg total air]
166 TYPE(ExtDat_3R), POINTER :: TK      ! Air temperature [K]
167 TYPE(ExtDat_3R), POINTER :: AIR     ! Dry air mass [kg]
168 TYPE(ExtDat_3R), POINTER :: AIRVOL  ! Air volume [m3]
169 TYPE(ExtDat_3R), POINTER :: AIRDEN  ! Dry air density [kg/m3]
170 TYPE(ExtDat_3R), POINTER :: O3      ! O3 mass [kg/kg dry air]
171 TYPE(ExtDat_3R), POINTER :: NO      ! NO mass [kg/kg dry air]
172 TYPE(ExtDat_3R), POINTER :: NO2    ! NO2 mass [kg/kg dry air]
173 TYPE(ExtDat_3R), POINTER :: HNO3   ! HNO3 mass [kg/kg dry air]
174 TYPE(ExtDat_3R), POINTER :: POPG   ! POPG mass [kg/kg dry air]
```

How to describe “Met Fields” for *HEMCO Extensions*?

- Plan 1: “Keep as is”
  - Use GEOS-FP met field format
  - Translate between “model native” & GEOS-FP using couplers
    - ✓ “Compatibility” with all extensions out-of-the-box
    - ✗ How to handle code duplication?
- Plan 2: Don’t impose extension met field format
  - Need to create a more flexible “registry”
    - ✓ Avoids mess of “state translators” (or use “CCPP”)
    - ✗ Cannot share extension sources between models

... other ideas welcome!

## Discussion: where(when) should emissions be added?

From Thibaud:

*Regarding the separation of HEMCO and mixing/drydep, I think that the most convenient way would probably be to add a field to State\_Chm that would hold tendencies (State\_Chm%Tendencies).*

- In GCC (and probably any other coupled model that uses HEMCO as a submodule?), this array would be filled through HEMCO as it is now. However it would be done outside of DO\_TEND/DO\_PBL\_MIXING.*
- In CESM-GC (where HEMCO lives at a higher level), the array could be populated in the interface (pp\_geoschem/chemistry.F90) through calls to the physics buffer.*

# Final Caveats

- Vertical regridding not working!
  - HCO\_Interp\_Mod only does 47 <-> 72 levels.
  - Should be able to do linear interpolation in pressure.
  - Will implement a lightweight "REGRID\_Z2Z\_Mod" to embed within HEMCO.
- Species list handling

The HEMCO-CESM project is made possible by:

- GCST and CESM® engineers and the GEOS-Chem ecosystem
- The CAM ionosphere interface
- The CESM2-GC project
- The ESMF framework