NEMSGSM

Building Global Spectral model in common infrastructure

Jun Wang

Overview

- NEMS GSM architecture
- Dynamics and physics grid component
- Learning from NEMSGSM to build earth science system:
  - Object Oriented design
  - Test system
  - Challenges in building NEMSGSM

ESMF based NEMS GSM- object oriented design

- NEMS (NOAA Environmental Modeling System) is a common modeling framework. It is based on ESMF (Earth System Modeling Framework) and follows NUOPC (The National NEMS Unified Operational Prediction Capability) convention

- A numerical model in NEMS is represented by software and implemented as an ESMF grid component.

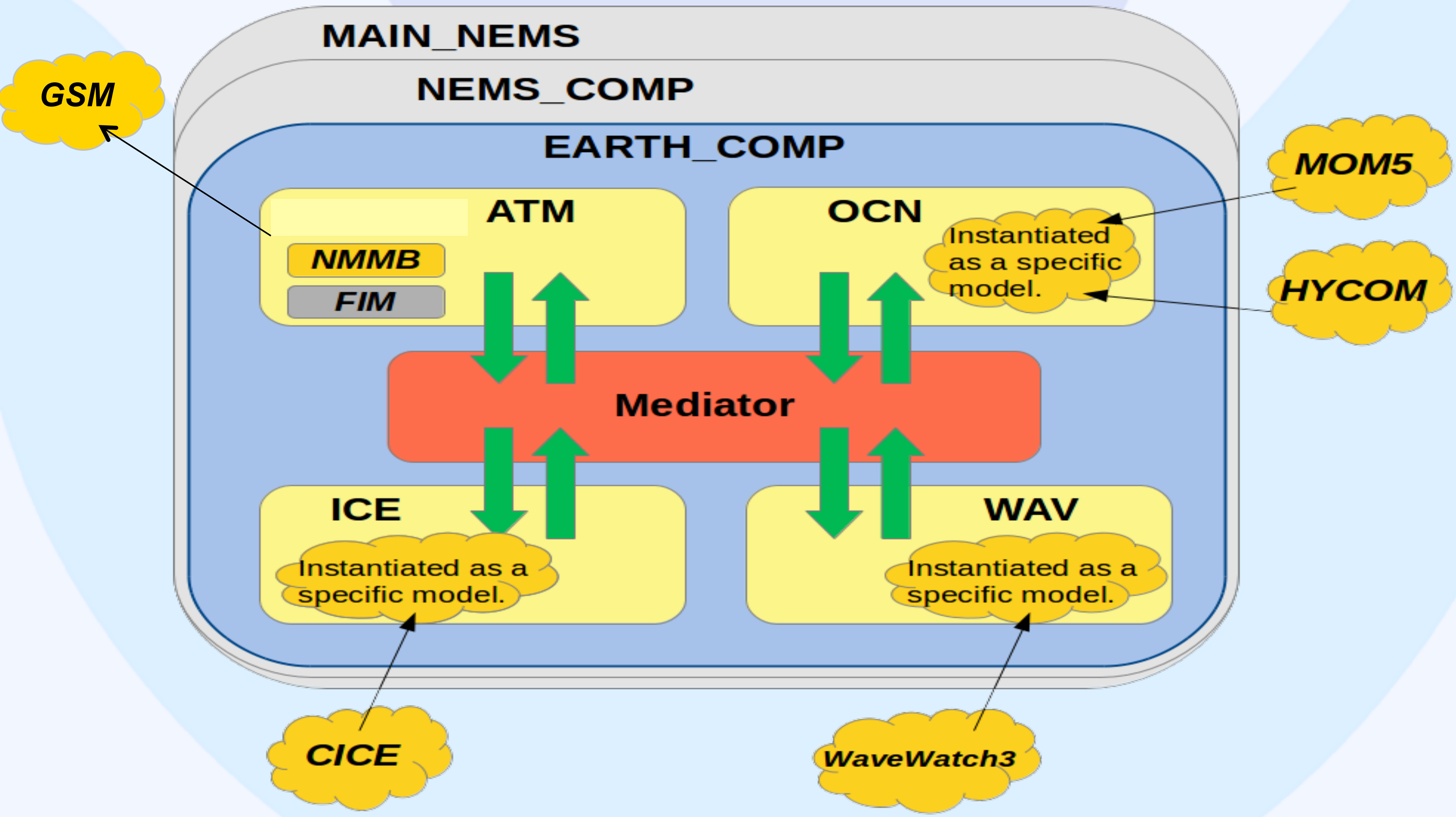
- Each ESMF grid component has import state and export state, they are the interface for inter-grid components communication. ESMF fields with decomposed domain grid information, are elements the import and export state for each component

- Each ESMF grid components has its own internal state with internal methods

NEMS atmosphere grid component

- The atmosphere component is the first component implemented in NEMS.
- In the early version of NEMS, the atmosphere component holds multiple atmosphere models such as GSM, NMMB and FIM. Currently GSM is an instantiation of atmosphere grid component, this makes the atmosphere grid component consistent with other components in NEMS system.
- Besides standalone atmosphere models, many coupled systems that consist of different earth science system components are currently under development in NEMS system.
- GSM is running in both standalone mode and coupler mode inside NEMS.

NEMS coupled system



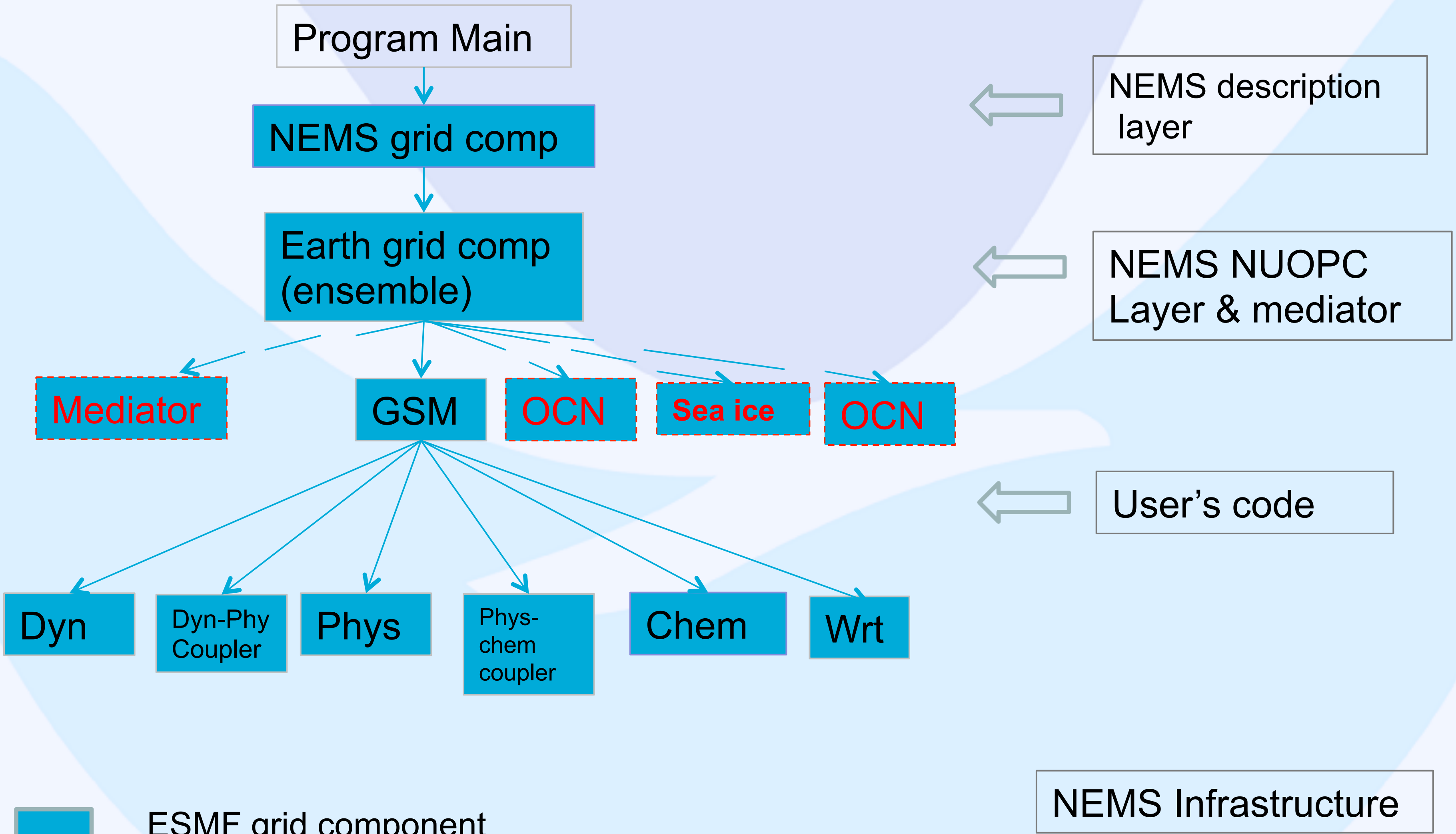
NEMSGSM development history

- Major milestones of NEMS GSM application development:
  - Eulerian dycore ~2007
  - Eulerian dycore + phys ~2008
  - Write grid component ~2009
  - Ensemble GSM ~2010
  - Inline POST ~2011
  - Whole Atmosphere Model (WAM) ~2012
  - NEMS GFS Aerosol Component (NGAC) ~2012
  - GFS Semi-Lagrangian dycore ~2014
  - GFS SL dycore + phys ~2015
  - UGCS seasonal ~2016
  - Physics updates for Q3FY2017 operational implementation ~2016

Major grid components in NEMS GSM

- Dynamics, physics, and model output are built as ESMF grid components
- Dynamics physics coupler is a ESMF coupler component located inside GSM, it is not going through NEMS mediator
- Chemistry grid component is ported from NASA/GSFC and connected into NEMSGSM through physics chemistry ESMF coupler component

NEMS GSM System



NEMSGSM NEMS driver

Program main

-ESMF initialize

-NEMS register (register NEMS initialize, run and finalize with ESMF)

-Register the NEMS component's Initialize, Run and finalize routines

-Create main clock

-Create NEMS grid comp import state, export state

-Execute NEMS comp initialization, run and finalize steps through standard ESMF interface

-ESMF finalize

NEMSGSM NEMS driver (cont.)

Module NEMS gridded component

-private data

-NEMS register (register NEMS initialize, run and finalize with ESMF)

-NEMS Initialization:

- Get internal state
- Create earth component and associated import state and export state for each ensemble member (and ensemble coupler for ensemble run)
- Register Initialize, Run, and Finalize subroutines for each earth component
- Call initialization of each earth component

-NEMS Run:

- Call run step of each earth component (ensemble coupler and then earth component)

-NEMS Finalize:

- Call finalize of each reach component

NEMSGSM NEMS driver (cont.)

Module earth gridded component:

* NUOPC layer, and earth subcomponents are selected through nems configure file

-private data

-Earth register (Inherit NUOPC driver, specialization NUOPC set Model services, run sequence, finalize and internal initialization; set up NUOPC field dictionary)

-Set model service: call setservice of selected components or mediator

-SetRunSequence: set run run sequences with selected components

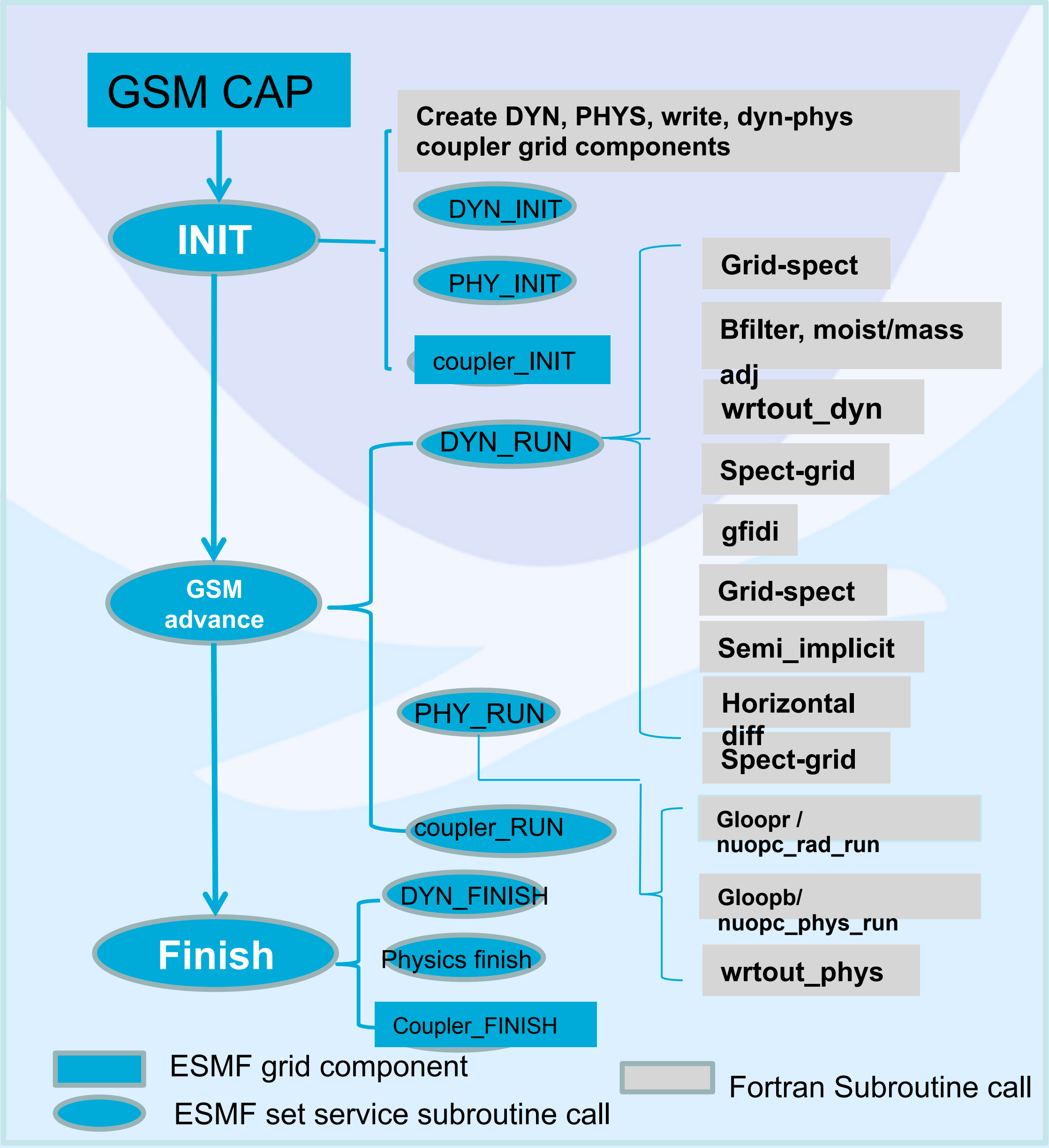
-Finalize: dealloate internal state

-Internal initialization: connector

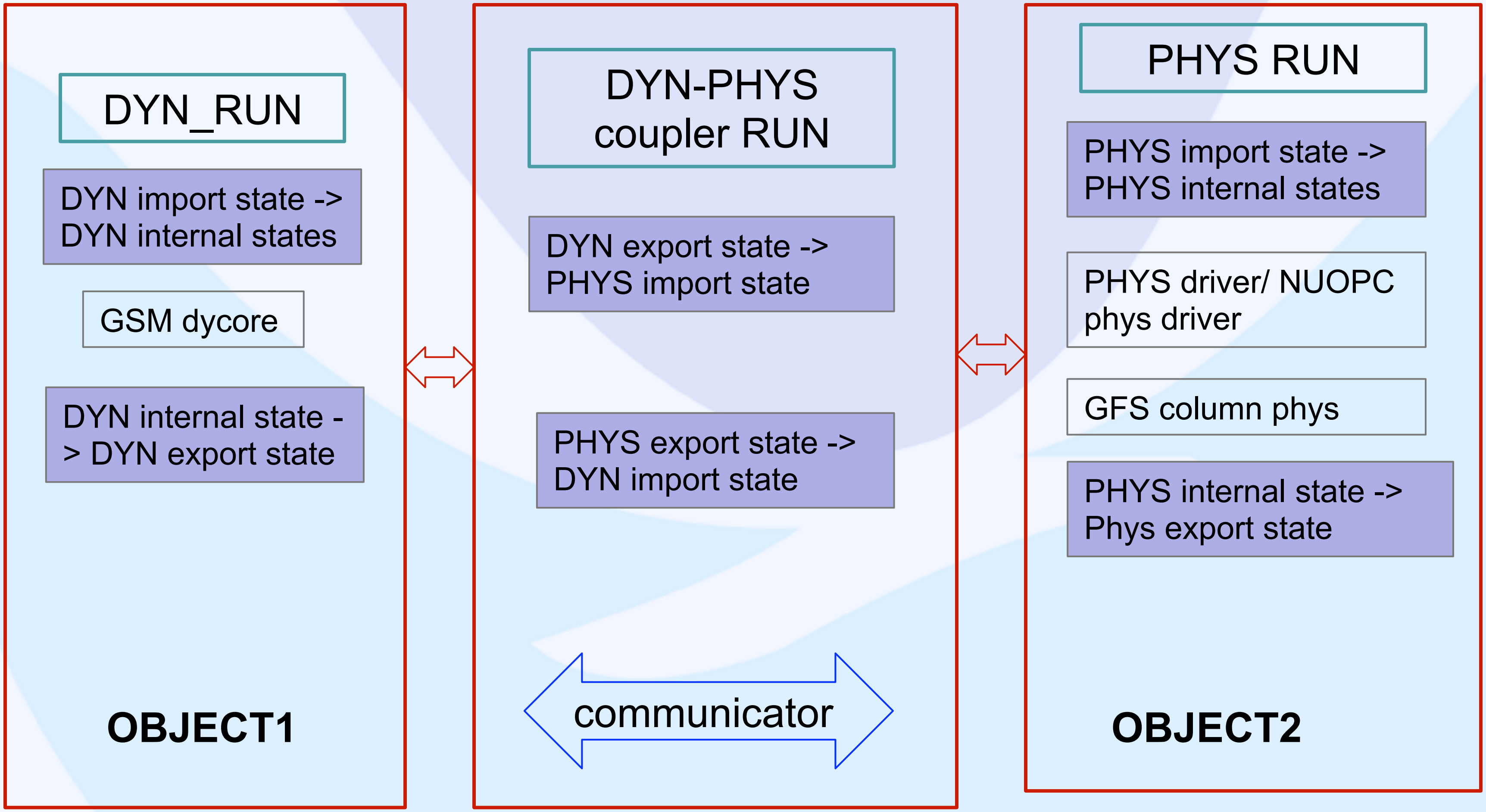
nems.configure:

```
EARTH_component_list: ATM
ATM_model:          ${atm_model:-gsm}
runSeq::
 ATM
::
```

NEMS GSM CAP flow chart



NEMS GSM Integration step



GSM Physics interface

- NEMS GSM dynamics and physics coupler
- GFS physics grid component
- Current GFS physics driver
- NUOPC physics driver

GFS Physics grid component

- gfs_phy_initialize
  - Get configure file variables, nest write import state in phys export state, get time/clock, call gfs physics related initialization routine, set up ESMF grid (currently Gaussian reduced grid) and ESMF output fields
- gfs_phy_run
  - Get time/clock, call phys run
- gfs_phy_finalize
  - Release memory

Current GFS phys driver

- Main data types in GFS physics driver:
  - grid_fld: model grid point data, z,ps, u,v,t, tracers, 3d stochastic physics weights, etc
  - g3d_fld: 3D atmospheric diag needed by GOCART, fcld dqdt, cnv_mfc. etc
  - aoi_fld: fields at atmosphere ocean interface, dusfc,dvsfc,dlwsfc,dswsfc,dnirbm,dnirbm,dnirdf…
  - sfc_fld: sfc property fields, tsea,   weasd, sncovr, tg3,   zorl, etc
  - flx_fld: derived and diagonostic field, SFCDSW, COSZEN, TMPMIN,   TMPMAX, SPFHMIN, DLWSFC

GFS radiation driver

```
CALL GLOOPR (grid_fld, g3d_fld, aoi_fld, lats_nodes_r
    &,            GLOBAL_LATS_R, LONSPERLAR, phyhour
    &,            deltim, XLON, XLAT, COSZDG, flx_fld%COSZEN
    &,            sfc_fld%SLMSK,  sfc_fld%weasd,  sfc_fld%SNCOVR
    &,            sfc_fld%SNOALB, sfc_fld%ZORL,   sfc_fld%TSEA
    &,            HPRIME, SFALB,  sfc_fld%ALVSF,  sfc_fld%ALNSF
    &,            sfc_fld%ALVWF,  sfc_fld%ALNWF,  sfc_fld%FACSF
    &,            sfc_fld%FACWF,  sfc_fld%CV,     sfc_fld%CVT
    &,            sfc_fld%CVB, SWH, SWHC, HLW, HLWC, flx_fld%SFCNSW
    &,            flx_fld%SFCDLW, sfc_fld%FICE,   sfc_fld%TISFC
    &,            flx_fld%SFCDSW, flx_fld%sfcemis

    &,            flx_fld%TSFLW,  FLUXR, phy_f3d, phy_f2d
    &,            SLAG, SDEC, CDEC, NBLCK, KDT, mdl_parm
!   &,            HTRSWB,HTRLWB      !idea add by hmhj
    &             )
```

GFS physics driver

```
call gloopb (grid_fld,     g3d_fld,       sfc_fld,
   &                 flx_fld,      aoi_fld,       nst_fld,
   &                 lats_nodes_r, global_lats_r, lonsperlar,
   &                 phydt,        phyhour,       sfalb,  xlon,
   &                 swh,          swhc,          hlw,    hlwc,
  &                  hprime,       slag,          sdec,   cdec,
   &                 ozplin,       jindx1,        jindx2,   ddy,
   &                 h2oplin,      jindx1_h,      jindx2_h, ddy_h,
   &                 phy_f3d,      phy_f2d,       phy_fctd, nctp,
   &                 xlat,         nblck,  kdt,   restart_step,
   &                 mdl_parm)
```

NUOPC physics driver 2.0

- NUOPC physics driver 2.0 is used in NEMSGSM

```
! Derived Data Types
      public :: state_fields_in          ! basic inputs of radiation and physics parameters
      public :: state_fields_out         ! basic outputs from radiation and physics
      public :: sfc_properties           ! surface fields
      public :: diagnostics              ! fields typically only used for diagnostic output
      public :: cloud_properties         ! cloud data and parameters
      public :: radiation_tendencies     ! radiation data
      public :: model_parameters         ! non-changing model parameters - set once in
initialize
      public :: interface_fields         ! data used for coupling (e.g. land and ocean)
      public :: dynamic_parameters       ! data that changes with integration loop
      public :: tbd_ddt                  ! to be determined data that has not been pigeonholed
```

NUOPC physics driver

- Function calls:

```
!! Main Subroutines
      public :: nuopc_phys_init        ! initialize routine
      public :: nuopc_phys_run         ! wrapper for gbphys
      public :: nuopc_rad_run          ! wrapper for grrad
      public :: nuopc_rad_update       ! wrapper for radupdate - updates some fields
between timesteps
      public :: nuopc_sppt_phys        ! stochastic physics
      public :: ozoneini
      public :: h2oini
```

NEMS GSM standalone application

- NEMS application NEMSlegacy is the NEMS GSM standalone repository
  - https://svnemc.ncep.noaa.gov/projects/nems/apps/NEMSGSM

- NEMS https://svnemc.ncep.noaa.gov/projects/nems/trunk

- GSM
- https://svnemc.ncep.noaa.gov/projects/gsm/trunk

- Chem
- https://svnemc.ncep.noaa.gov/projects/aerosol/chem/trunk

GFS Physics repository

- GFS physics is located at:

https://svnemc.ncep.noaa.gov/trac/gsm/gsmphys

- NEMSGFS physics:
  - Phys: gsm physics wrapper
  - Gsmphys: gfs column physics

PHYS RUN

PHYS import state -> PHYS internal states

PHYS driver/ NUOPC phys driver

GFS column phys

PHYS internal state -> Phys export state

Features of NEMSGSM:

ESMF based NEMS GSM- object oriented design

- The basic concept of the ESMF is that complicated applications can be broken up into coherent pieces with standard calling interface.
- The earth science system subcomponents are implemented as cohesive objects with associated internal state and methods combined hidden inside the grid component.
- The encapsulation can provide data protection and reduce system complexity, and thus increase system robustness. This is different from traditional procedural system that model data and procedure separately.
- It is also useful design for interoperatable physics driver.

Features of NEMSGSM: test system

- NEMSGSM test system provides evidence for accept new code changes
  - Regression test
    - Decomposition, threading, resolution, regional/global
    - Evolve: add test, update setting for all test after implementation
  - Parallel test for operational implementation
  - A unified workflow system is under development, but testing NEMSGSM as the atmosphere component in many coupled applications remains a big challenge:
    - Review a certain code updates in weather and climate mode is a science topic
    - How to set up climatology from standalone NEMSGSM that evolves fast
    - Add test cases for special events

Challenges in working with NEMSGSM

- Evolve fast
  - Ticket system: document all the code changes
  - Merge to trunk as frequently as possible
  - Break big task into small pieces and make several commits

- Complexity
  - Many components, It is impossible to understand all the pieces
  - Communication is important

Thanks!