A comparative study of coupling frameworks: the MOM case study

V. Balaji

Princeton University and NOAA/GFDL

Giang Nong and Shep Smithline

RSIS Inc. and NOAA/GFDL

Rene Redler

NEC Europe Ltd

ECMWF High Performance Computing Workshop

Reading, UK

25 October 2004

Overview of talk

- Emergence of coupling frameworks
- Current capabilities at GFDL: the FMS coupler
- Implementation in ESMF and PRISM
- Towards a layered approach

Technological trends

In climate research... increased emphasis on detailed representation of individual physical processes governing the climate; requires many teams of specialists to be able to contribute components to an overall coupled system;

In computing technology... increase in hardware and software complexity in HPC, as we shift toward the use of scalable computing architectures.

In software design for broad communities... The open source community provided a viable approach to the construction of software to meet diverse requirements through “open standards”. The standards evolve through consultation and prototyping across the user community.

The community response: modernization of modeling software

- Abstraction of underlying hardware to provide ***uniform programming model*** across vector, uniprocessor and scalable architectures;

- Distributed development model: many contributing authors. Use high-level abstract language features to facilitate development process;

- Modular design for interchangeable dynamical cores and physical parameterizations, development of ***community-wide standards*** for components.

FMS and MOM

GFDL developed the Flexible Modeling System (FMS) starting in 1997, as a means of unifying all of GFDL models within a common codebase and shared software infrastructure. It is a design prototype for the emerging community coupling frameworks: Earth System Modeling Framework (ESMF) and PRogramme for Integrated earth System Modeling (PRISM).

The GFDL Modular Ocean Model (MOM) is a publicly available ocean model distributed with FMS. It is a Z-coordinate ocean model in generalized curvilinear horizontal coordinates, with an explicit free surface, a non-Boussinesq option, and 2- and 3- time-level schems. It has been used in studies spanning a wide range of time and space scales, from short-term studies of channel flows to coupled climate integrations on 1000-year scales. A wide range of physics packages is available for use at various resolutions.

Links:

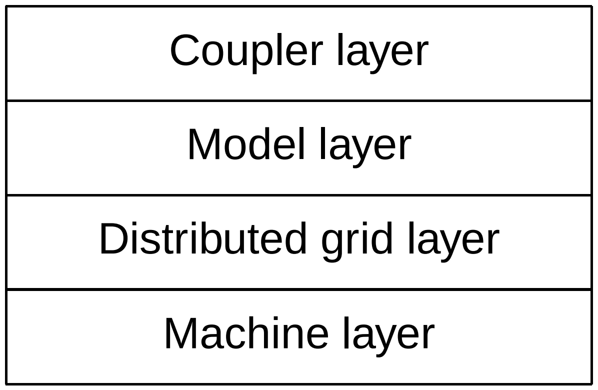
FMS and MOM `http://www.gfdl.noaa.gov/ fms`

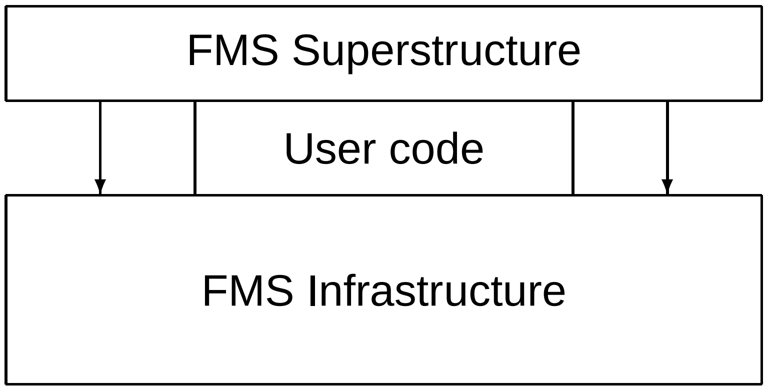
Downloads: `http://fms.gfdl.noaa.gov`

ESMF `http://www.esmf..ucar.edu`

PRISM `http://prism.enes.org`

Architecture of FMS





FMS shared infrastructure: machine and grid layers

MPP modules communication kernels, domain decomposition and update, parallel I/O.

Time and calendar manager tracking of model time, scheduling of events based on model time.

Diagnostics manager Runtime output of model fields.

Data override Runtime input of model fields.

Scientific libraries Uniform interface to proprietary and open scientific library routines.

The FMS coupler

Used for data exchange between models. Key features include:

Conservation: required for long runs.

Resolution: no constraints on component model timesteps and spatial grid. Supports both explicit and implicit timestepping.

Exchange grid: union of component model grids, where detailed flux computations are performed (Monin-Obukhov, tridiagonal solver for implicit diffusion, ...)

Fully parallel: Calls are entirely processor-local: exchange software will perform all inter-processor communication.

Modular design: uniform interface to main calling program.

No brokering: each experiment must explicitly set up field pairs.

Single executable:

Highly efficient: currently able to couple atmos/ocean explicitly at each ocean timestep, atmos/land/ice implicitly at each atmos timestep for current dec/cen models.

Implicit coupling and the exchange grid

Fluxes at the surface often need to be treated using an implicit timestep. (e.g temperature flux in near-surface layers that can have vanishingly small heat capacity.) This feature is key in the design of the FMS coupler. Consider simple vertical diffusion in a coupled atmosphere-land system:

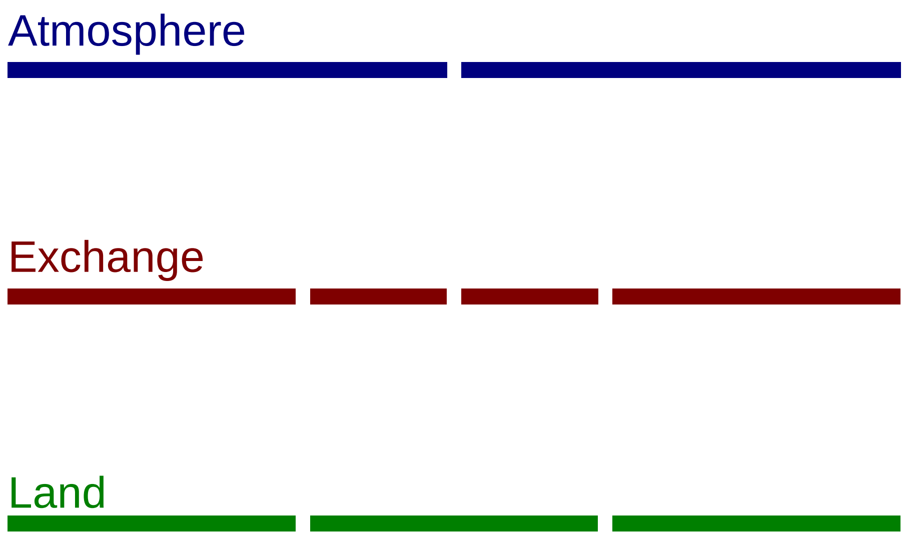
$$\frac{\partial T}{\partial t} = -K\frac{\partial^2 T}{\partial z^2} \tag{1a}$$

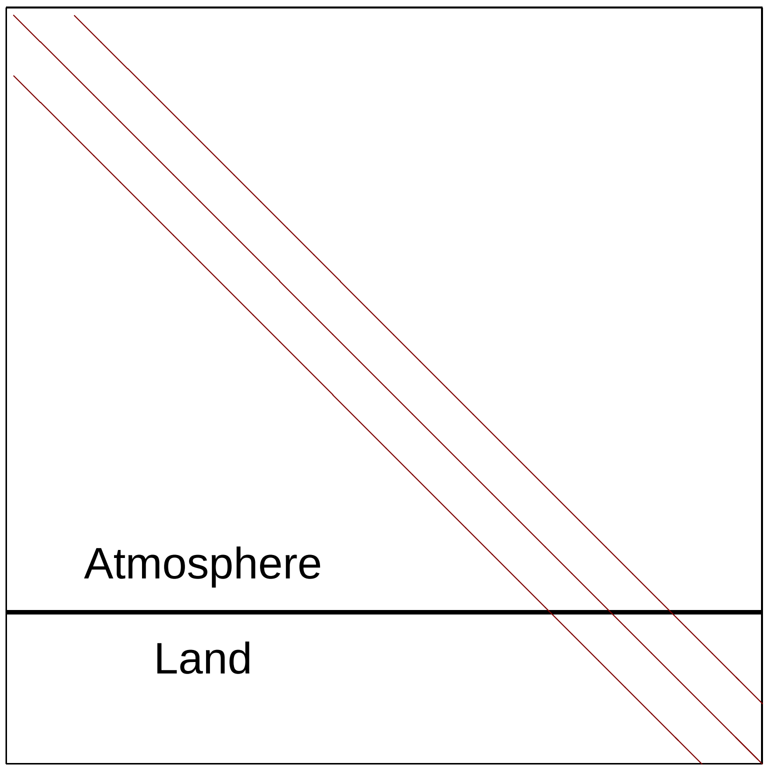
$$\frac{T_k^{n+1} - T_k^n}{\Delta t} = -K\frac{T_{k+1}^{n+1} + T_{k-1}^{n+1} - 2T_k^{n+1}}{\Delta z^2} \tag{1b}$$

$$\mathbf{A}\mathbf{T}^{n+1} = \mathbf{T}^n \tag{1c}$$

This is a tridiagonal matrix inversion which can be solved relatively efficiently using an up-down sweep. The problem is that some of the layers are the atmosphere and others are in the land. Moreover the components may be on different grids.

Implicit coupling and the exchange grid





coupler main loop

```
do nc = 1, num_cpld_calls
   call generate_sfc_xgrid( Land, Ice )
   call flux_ocean_to_ice( Ocean, Ice, Ocean_ice_flux )
   call update_ice_model_slow_up( Ocean_ice_flux, Ice )
!fast loop
   call update_land_model_slow(Land)
   call flux_land_to_ice( Land, Ice, Land_ice_flux )
   call update_ice_model_slow_dn( Atmos_ice_flux, Land_ice_flux, Ice )
   call flux_ice_to_ocean( Ice, Ice_ocean_flux )
   call update_ocean_model( Ice_ocean_flux, Ocean )
enddo
```

(2)

Ocean boundary

The boundary state is the state of the ocean model visible outside: the fluxes are the fields it needs for its boundary forcing.

```
type ocean_boundary_data_type
   type(domain2D) :: Domain
   real, pointer, dimension(:,:) :: t_surf, s_surf, sea_lev, &
        frazil, u_surf, v_surf
   logical, pointer, dimension(:,:) :: mask
   type (time_type)                 :: Time, Time_step
end type ocean_boundary_data_type

type, public :: ice_ocean_boundary_type
   real, dimension(:,:), pointer :: u_flux, v_flux, t_flux, q_flux
   real, dimension(:,:), pointer :: salt_flux, lw_flux, sw_flux, lprec, fprec
   real, dimension(:,:), pointer :: runoff, calving
   real, dimension(:,:), pointer :: p
   real, dimension(:,:,:), pointer :: data
   integer :: xtype             !REGRID, REDIST or DIRECT
end type ice_ocean_boundary_type
```

Flux exchange

Three types of flux exchange are permitted: `REGRID`, `REDIST` and `DIRECT`.

REGRID physically distinct grids, requires exchange grid.

REDIST identical global grid, different domain decomposition.

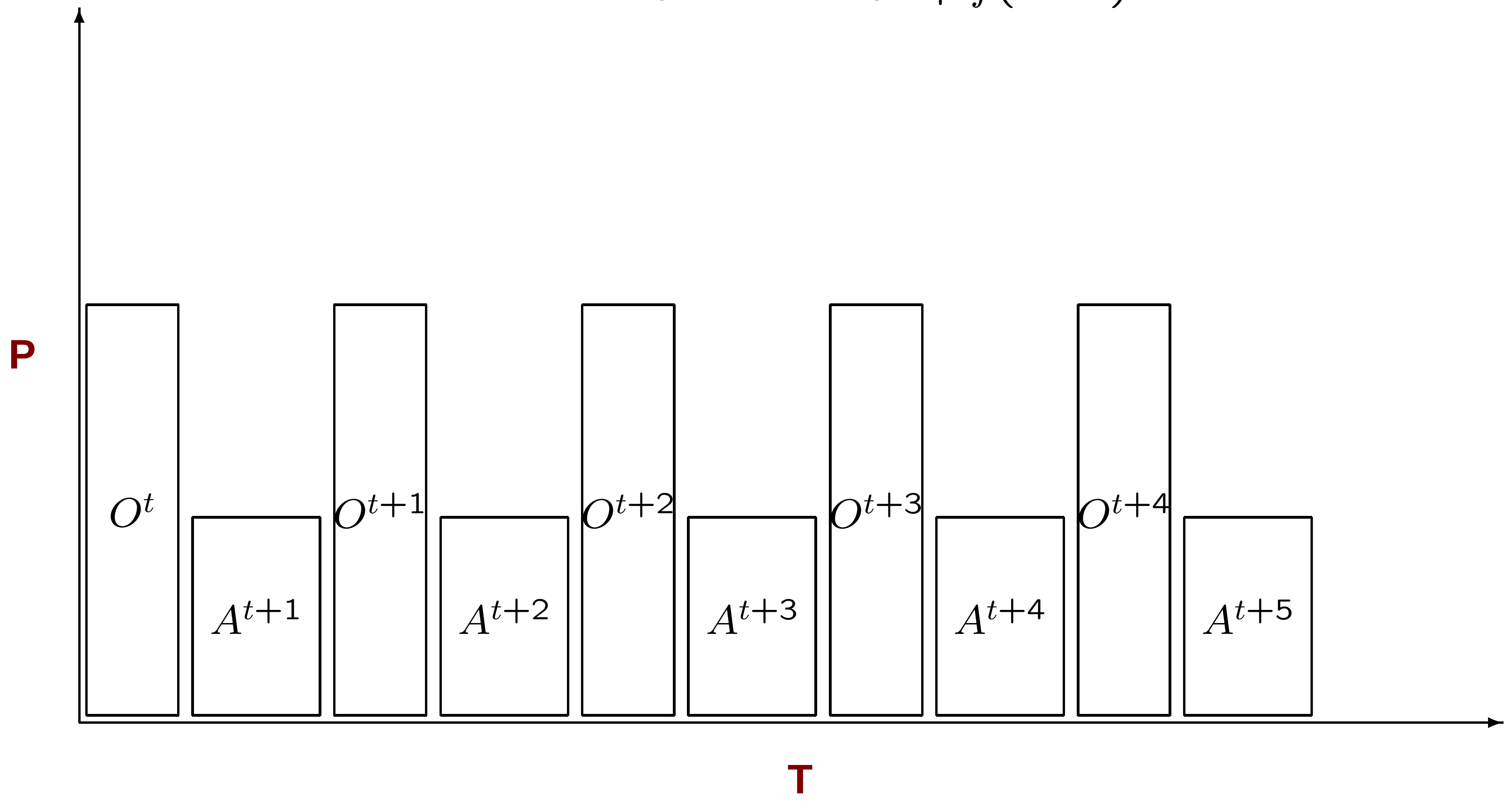
DIRECT identical grid and decomposition.

Current use: `REGRID` between atmos$\Longleftrightarrow$ice, atmos$\Longleftrightarrow$land, land$\Longleftrightarrow$ice, `REDIST` between ocean$\Longleftrightarrow$ice.

Serial coupling

Uses a forward-backward timestep for coupling.

$$A^{t+1} = A^t + f(O^t) \tag{3}$$
$$O^{t+1} = O^t + f(A^{t+1}) \tag{4}$$

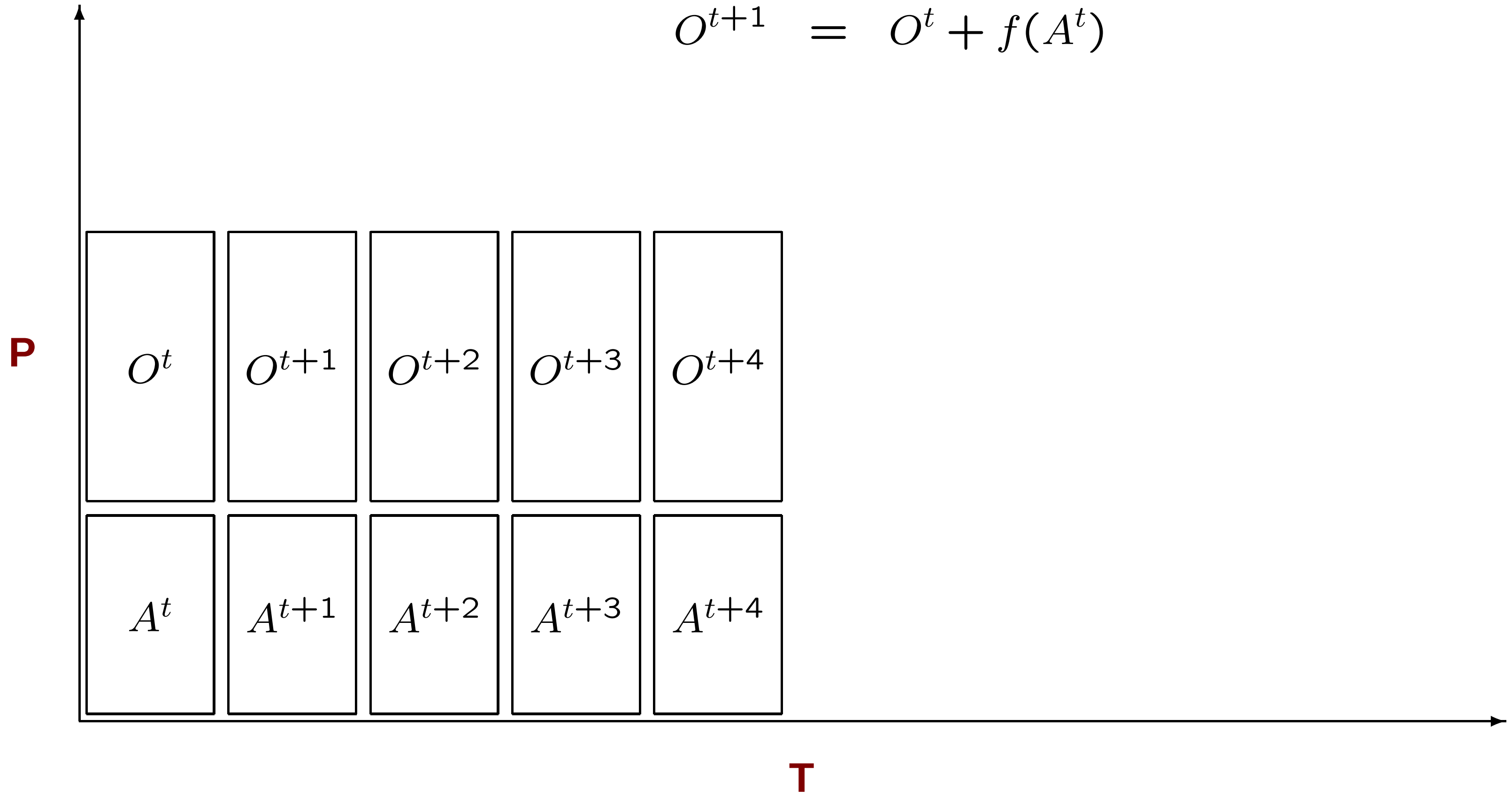


Concurrent coupling

This uses a forward-only timestep for coupling. While formally this is unconditionally unstable, the system is strongly damped. Answers vary with respect to serial coupling, as the ocean is now forced by atmospheric state from Δt ago.

$$A^{t+1} = A^t + f(O^t) \tag{5}$$

$$O^{t+1} = O^t + f(A^t) \tag{6}$$



In terms of model code...

```
do nc = 1, num_cpld_calls
   call generate_sfc_xgrid()
   call flux_ocean_to_ice()
   if( use_lag_fluxes )call flux_ice_to_ocean()
   if( atmos_pe )then
       call update_ice_model_slow_up()
       call update_atmos... !fast loop
       call update_land_model_slow()
       call flux_land_to_ice()
       call update_ice_model_slow_dn()
   endif
   if( .NOT.use_lag_fluxes )call flux_ice_to_ocean()
   if( ocean_pe )call update_ocean_model()
enddo
```

- The pelists are set up in the coupler layer, and subsequently all the **`mpp`** calls automatically operate within their pelists, with no changes to the model code.

- Within the atmos pelist, we can further declare land and ice as concurrent if needed. Not currently implemented, since $T_{\text{ice}} \gg T_{\text{land}}$.

Fitting into FMS

To incorporate your own ocean model (say) into FMS, you have to provide **`init`**/**`run`**/**`exit`** routines (**`ocean_model_init`**, **`update_ocean_model`**, **`ocean_model_end`**) and also encapsulate your ocean boundary state into **`ocean_boundary_type`**.

It helps to use the FMS infrastructure but not essential. For users with, say, a solo atmospheric model code that they wish to couple to MOM, the advantage is that MOM is distributed with the FMS coupler, and all that is needed is to write a few “wrapper” routines for the atmospheric model.

Features of the FMS coupler

- Predefined list of components (atmosphere, ocean, land, ice).
- Encapsulated boundary state and boundary fluxes with predefined fields. (There is provision for adding an arbitrary bundle of tracers at runtime).
- Support for serial and concurrent coupling within single executable.
- Implicit coupling between land-ocean surface and atmosphere on atmospheric timestep; explicit coupling between ocean surface and ocean on ocean timestep. The atmosphere run method is split into down-up phases for the tridiagonal solver.
- Coupler serves as the scheduler for components, which return control to the coupler at the end of an independent execution segment.
- Support for ensembles (multiple instances of the same component).

The ESMF coupler

The ESMF coupler closely follows the architecture of FMS, but with a more general and powerful notion of a component. A component is defined by

- its structure, consisting of an `init`, `run` and `exit` method. Unlike FMS, the coupler does not directly invoke these routines. These are registered for use by an `ESMF_SetServices()` call.
- its import and export states (analogous to the boundary fluxes and boundary states in FMS).
- a coupler must be written for every pair of components. This is not as onerous as it sounds: coupler functions are fairly generic and once one exists, other similar components can be plugged in with no effort at all. Efforts are underway to generalize this “middleware”.

Making a MOM ESMF component

- Register the **init**/**run**/**exit** methods.

```
type(ESMF_GridComp) ::  comp
...
call ESMF_GridCompSetEntryPoint(comp,ESMF_SETRUN,update_ocean_mode
```

(7)

- Create the import and export states from FMS datatypes. This can be done entirely with pointers: no data movement is involved.

```
ocean%u_surf=>expFMSArray(1)%ptr
...
expESMF_Field(i)=ESMF_FieldCreate(ocnGrid,expFMSArray(i)%ptr,...)
call ESMF_StateAddField(expState,expESMF_Field,...)
```

(8)

- Run the component.

```
type(ESMF_GridComp) ::  compOcn
...
compOcn=ESMF_GridCompCreate(vm,"Ocean",rc=rc)
call ESMF_GridCompSetServices(compOcn,OceanRegister,rc)
...
call ESMF_GridCompRun(compOcn,impOcn,expOcn,topClock,rc=rc)
```

(9)

The PRISM coupler

The PRISM coupler is designed with additional constraints: where components may not be able to be part of a common executable, where their execution sequence may not be easily parsed into **`init`**, **`run`**, and **`exit`** methods, and where components may not be able to return control to an external entity during execution. This has led to a different design, where concurrency is the norm, and components and the PRISM coupler may all be independent executables.

External configuration files (PMIOD/SMIOC/SCC) are used to share run-time configuration information between components and the PRISM system.

After configuration, components exchange data using **`PRISM_Put`** (non-blocking) and **`PRISM_Get`** (blocking) calls. All execution is concurrent, with the **`PRISM_Get`** block implicitly enforcing synchronization.

Making a MOM PRISM component

Write the SMIOC configuration information for MOM.

```
<transient local_name="ocean_SST">
    <standard_name>sea surface temperature</standard_name>
    <computation mask="true" mask_time_dependency="false"
                                       method_type="mean">
      <associated_gridfamily local_name="ocn_grid" />
      <associated_compute_space local_name="center" />
    </computation>

    <intent>
      <output transi_out_name="ocean_SST_out">
        <exchange_date>
          <period>
            <nbr_secs type="xs:integer">43200</nbr_secs>
          </period>
        </exchange_date>
        <corresp_transi_in type="xs:string">atm_SST_in</corresp_transi_
        <component_name type="xs:string">atm</component_name>
    </intent>
  </transient>
```

Making a MOM PRISM component

- Ingest the configuration information in **`ocean_model_init`**.

```
call prism_def_var ( var_id, var_name, grid_id, method_id, &
   mask_id, var_nodims, actual_shape, var_type, ierr )
```

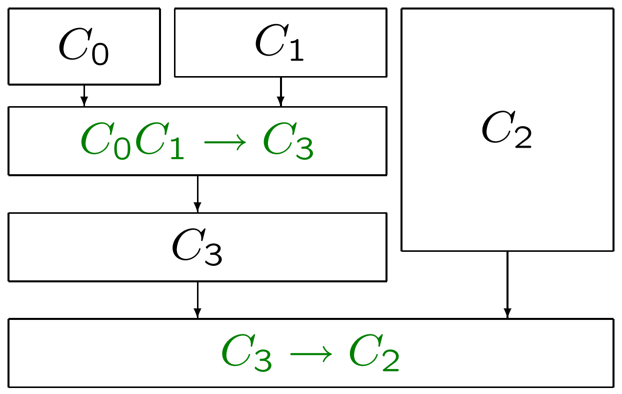
(10)

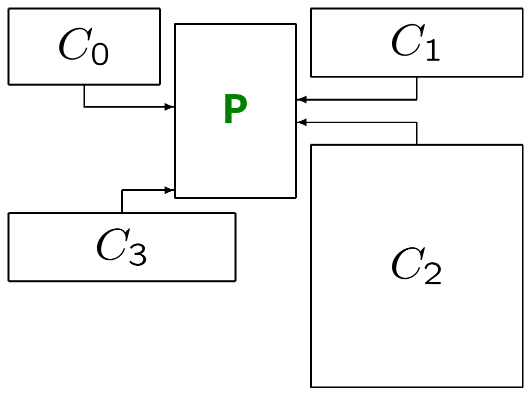
- Modify **`update_ocean_model`** to add the appropriate **`PRISM_Put`** and **`PRISM_Get`** calls.

```
call get_date (Time%model_time, date%year, date%month, &
   date%day, date%hour, date%minute, second)
call PRISM_calc_newdate ( ...  )
call PRISM_Put( var_id, date, date_bound, data, info, ierr )
```

(11)

Comparison of coupling models





- In FMS and ESMF, after each independent component run segment, control is returned to the coupler, which runs on the union of all PEs of its child components.

- PRISM uses a client-server model where all components execute concurrently, and the coupler **`P`** processes their **`PRISM_Put`** and **`PRISM_Get`** requests. Configuration of the coupler is through external files (SMIOC/SCC).

Conclusions: toward a layered approach

- In FMS and ESMF, the coupler is part of a ***superstructure***, which exercises control and scheduling functions for components. Components must be structured in terms of **`init`**, **`run`** and **`exit`** methods. In PRISM it is part of the ***infrastructure***, sharing configuration information and processing requests from components. This is the key architectural difference, not the MPMD/SPMD distinction. Synchronization in either architecture is provided by our applications, which remain in unison, never more than one coupling timestep out of synchronicity.

- The ESMF generic component interface, discussed in another talk, is a powerful abstraction for creating coupled models, in contrast with FMS, which is specifically targeted at coupled climate models with standard components. It is likely that a generic coupler with FMS-like structure and capabilities will emerge for coupled climate modeling, but based on ESMF components. This will be a significant community resource.

- The ESMF component and state data structures are rich in metadata, matching the information provided in PRISM configuration files. It is likely that a layered approach sharing configuration management standards will emerge shortly. Grid standards are an important emerging development in this area.

- Drivers for FMS/ESMF/PRISM can be automatically generated based upon this metadata.