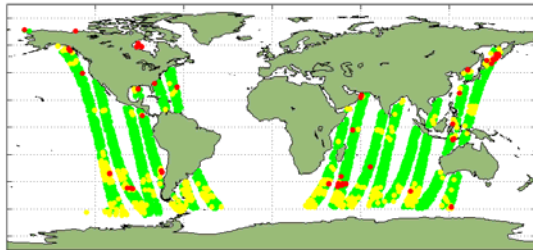


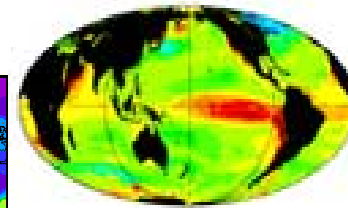
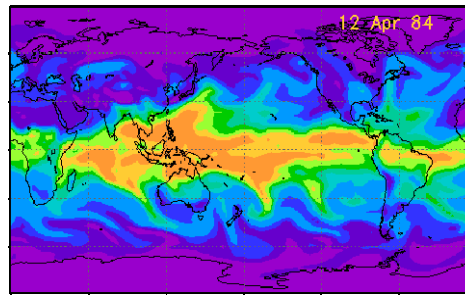
NSF NCAR / NASA GSFC / DOE LANL ANL / NOAA NCEP GFDL / MIT / U MICH

“Analyze This” – The Earth System Modeling Framework’s (ESMF) component model

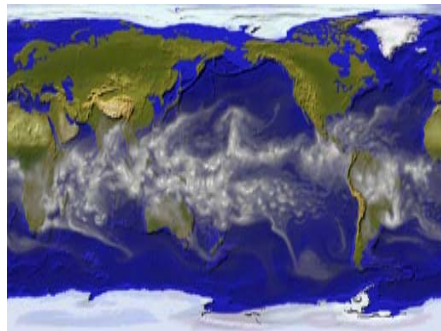


NASA GMAO PSAS

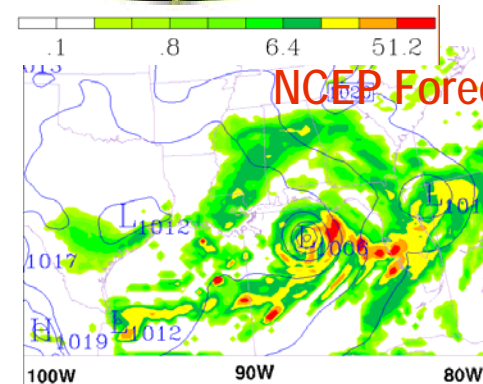
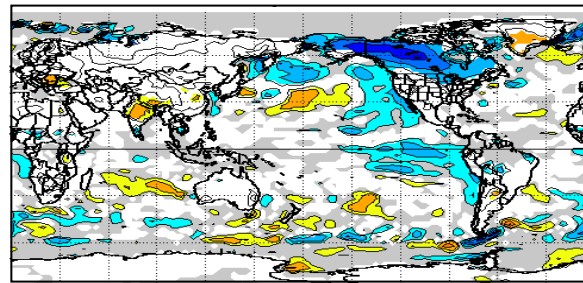
GFDL FMS Suite



MITgcm



NCAR/LANL CCSM

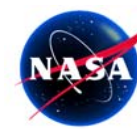


NCEP Forecast

NASA GMAO Seasonal Forecast

Chris Hill, MIT cnh@plume.mit.edu

AD Workshop – Large Scale Apps, 2005, Shrivenham, UK



Talk Outline

ESMF

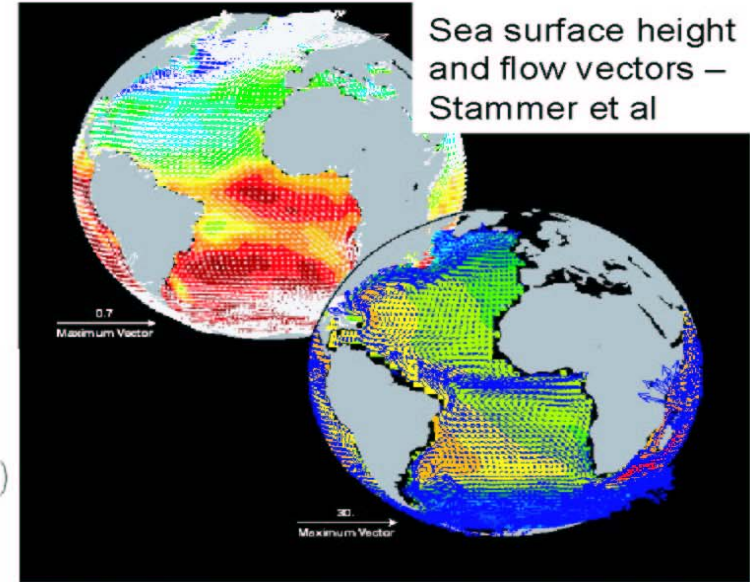
- **Background**
 - AD in Earth system modeling
 - ESMF goals
- **The ESMF Component Model**
 - Component hierarchies
 - Dataflow between components
 - Controlling components
- **Under the hood**
- **AD on this sort of large-scale, multi-component application**



I) Model data synthesis

- ECCO uses DA to estimate full depth ocean state on interannual, decadal time scales.
 - DA is adjoint based 4dvar, employs automatic differentiation (AD) to generate adjoint
 - Cost function is 10 year misfit from a spectrum of ocean observations

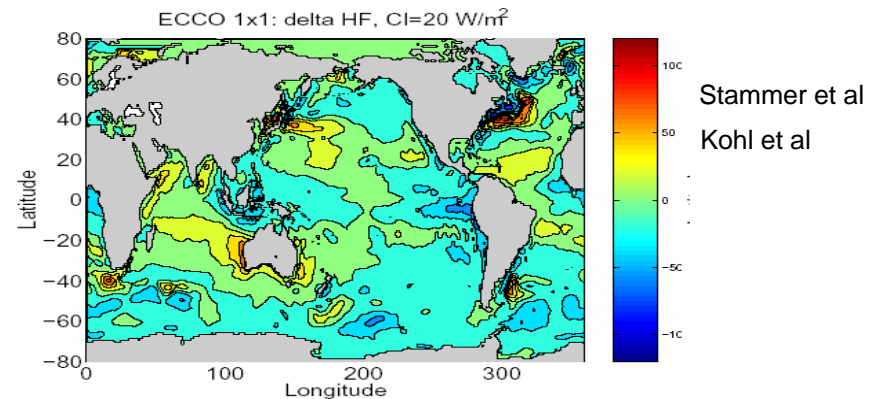
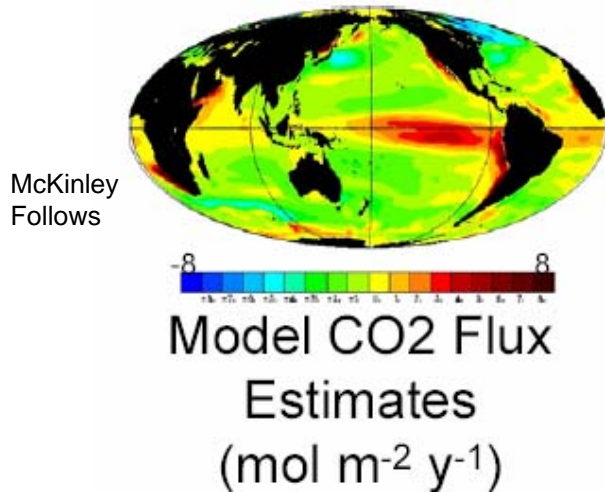
$$\begin{aligned}
 J = \frac{1}{2} & \left[\left(\bar{\eta} - \bar{\eta}_{lp} \right)^T W_{\text{geoid}} \left(\bar{\eta} - \bar{\eta}_{lp} \right) \right. \\
 & + \left(\eta' - \eta'_{lp} \right)^T W_{\eta_{lp}} \left(\eta' - \eta'_{lp} \right) + \left(\eta' - \eta'_{ers} \right)^T W_{\eta_{ers}} \left(\eta' - \eta'_{ers} \right) \\
 & + \left(\delta \tau_x \right)^T W_{\tau_x} \left(\delta \tau_x \right) + \left(\delta \tau_y \right)^T W_{\tau_y} \left(\delta \tau_y \right) \\
 & + \left(\delta H_Q \right)^T W_{H_Q} \left(\delta H_Q \right) + \left(\delta H_F \right)^T W_{H_F} \left(\delta H_F \right) \\
 & + \left(\delta T_0 \right)^T W_T \left(\delta T_0 \right) + \left(\delta S_0 \right)^T W_S \left(\delta S_0 \right) \\
 & + \sum_i \left(\bar{\theta}_i - \bar{\theta}_{iSST} \right)^T W_{SST} \left(\bar{\theta}_i - \bar{\theta}_{iSST} \right) \\
 & + \sum_i \left(\bar{\theta}_i - \bar{\theta}_{iLevi} \right)^T W_T \left(\bar{\theta}_i - \bar{\theta}_{iLevi} \right) \\
 & + \sum_i \left(\bar{S}_i - \bar{S}_{iLevi} \right)^T W_S \left(\bar{S}_i - \bar{S}_{iLevi} \right) \left. \right].
 \end{aligned}$$



Sea surface height and flow vectors – Stammer et al

The estimated mean sea-surface height and velocity field at 27 m depth are shown in the lower part of the figure. The figure shows all major current systems, but with low model resolution they appear too smooth. The upper part shows an instantaneous field of wind stress and net heat flux that is being used to force the model. The time-varying forcing is a control variable that is being estimated to bring the model into consistency with the data. Substantial corrections to the NCEP surface heat flux fields used as first guess are required. Changes relative to NCEP estimates are within error bounds and are in the range of +/- 20 W/m² over large parts of the ocean. Over boundary currents changes can reach +/- 80 W/m² but are still consistent with our understanding of NCEP heat flux errors.

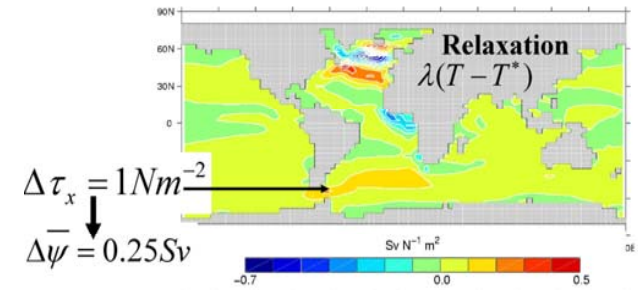
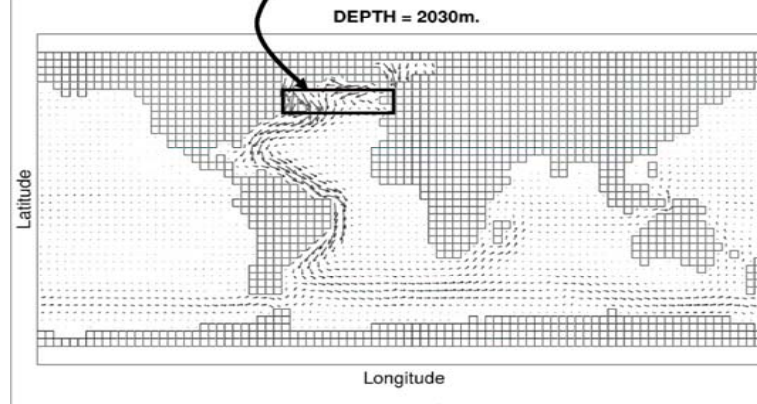
Used to make estimates of planetary scale phenomena on decadal time scales



II) Sensitivity Analysis

MOC Sensitivity to Wind Stress

$$J = \bar{\psi} \left(\begin{array}{l} \phi = 60N - 64N, \\ z = -1055m, -1785m \end{array} \right)$$

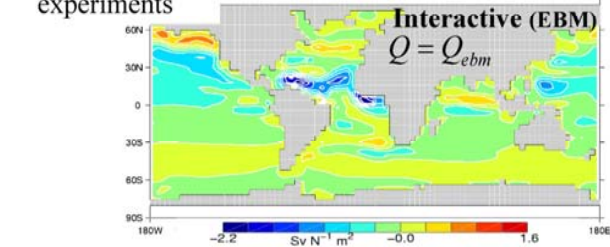


Results

$$\frac{\partial \bar{\psi}}{\partial \tau_x}$$

$$= 10^4 \text{ perturbation experiments}$$

experiments



$$\frac{\partial C}{\partial t} = -\vec{U} \cdot \nabla C + \nabla \cdot (K \nabla C) + \Gamma(C) - \mu C + S$$

Tracer equation

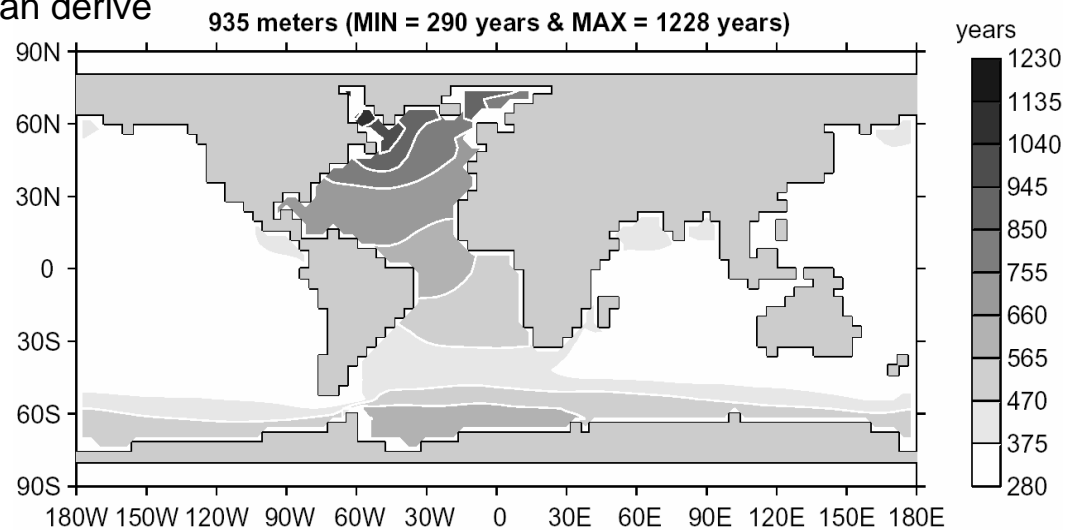
$$J(t = T) = \int_{t=0}^{t=T} \int_A \mu C \Delta z dA dt$$

Using $\frac{\partial J}{\partial S}(\lambda, \phi, z, t)$ we can derive

Air-sea exchange cost function

Analyzing model air-sea gas exchange of a tracer subject to time-dependent transport

global maps of mean residence time for a tracer injection in the ocean. Adjoint computational work is approx. 8000 years simulated time. Non-adjoint would require 108 million simulated years!



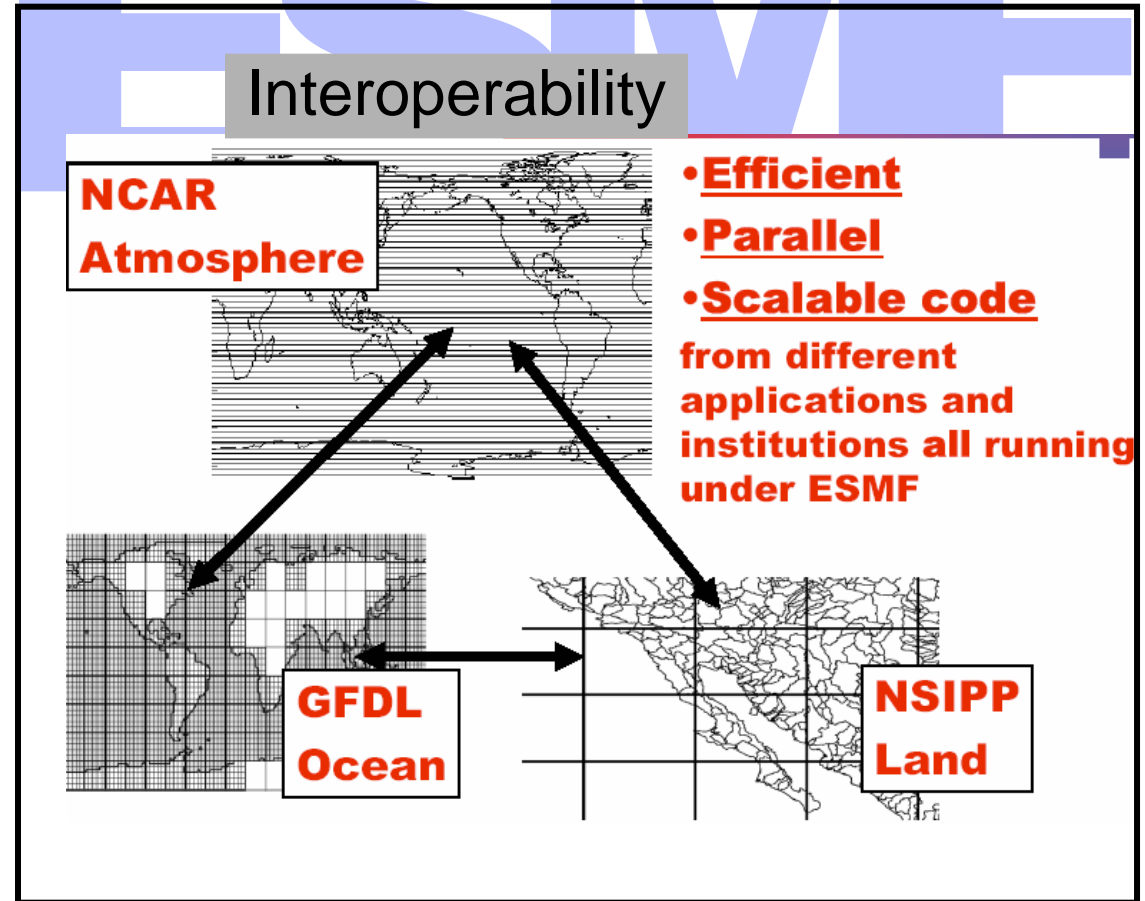
NSF NCAR / NASA GSFC / DOE LANL ANL / NOAA NCEP GFDL / MIT / U MICH

ESMF Goals

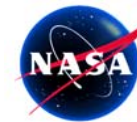
GOAL: Build a software superstructure and infrastructure that supports technical interoperability and sharing of code between Earth system modeling groups.

PRODUCTS:

- Coupling superstructure and utility infrastructure software



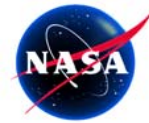
Achieving interoperability requires some commonality in programming models – in ESMF, it is the component model that provides commonality.



Talk Outline

ESMF

- Background
 - AD in Earth system modeling
 - ESMF goals
- The ESMF Component Model
 - Component hierarchies
 - Dataflow between components
 - Controlling components
- Under the hood
- AD on this sort of large-scale, multi-component application



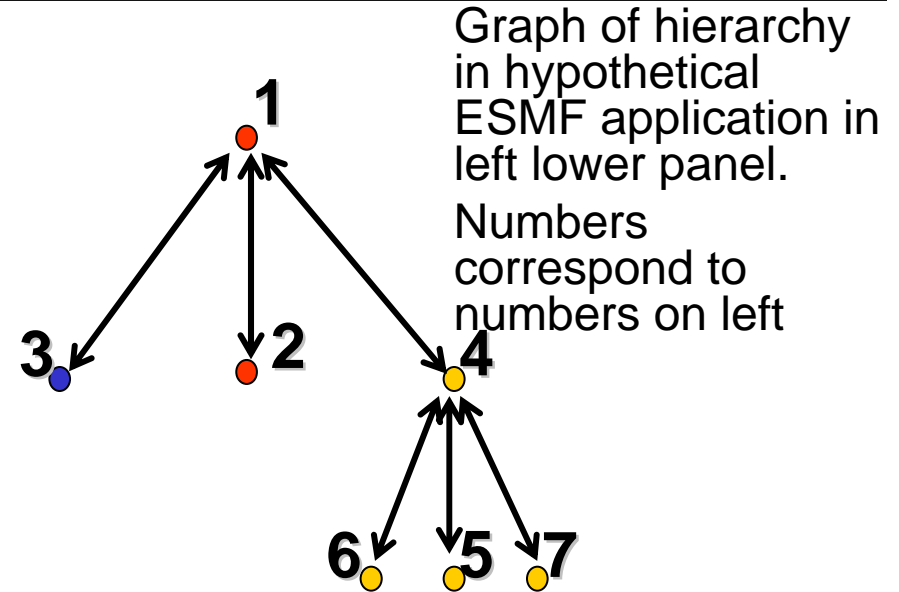
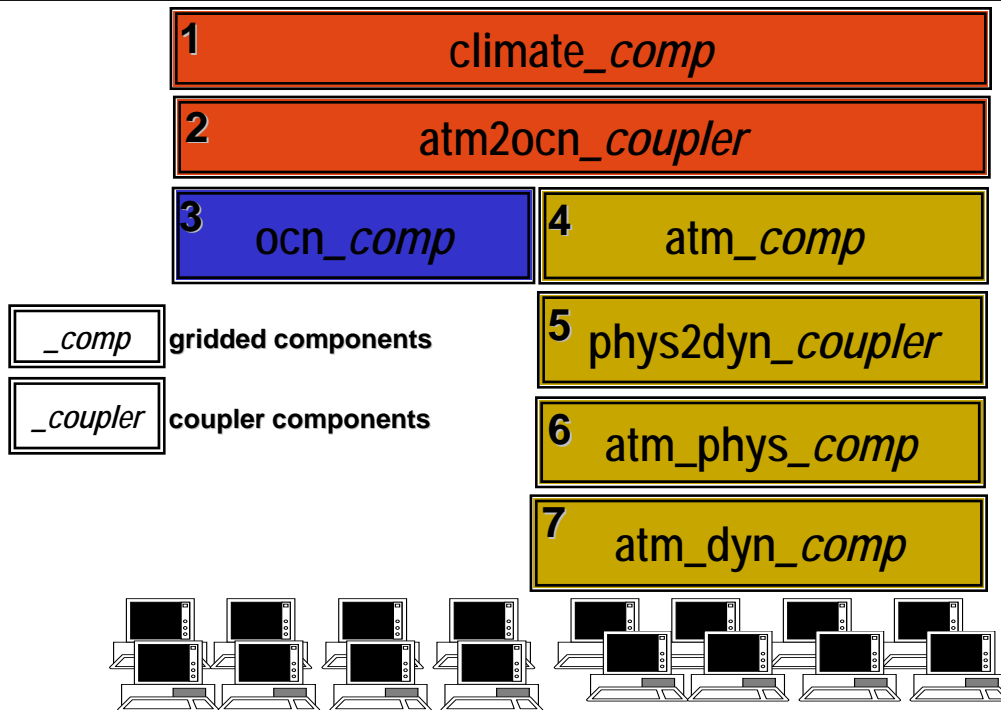
The ESMF Component Model

Provides machinery to code an ESMF application == a hierarchy of interacting

- coupler components, e.g. *ocm2atm_coupler* and
- gridded components, e.g. *atm_phys_comp*

Machinery includes

- general-purpose mechanisms for developers to code “wirings” between components (*ESMF_State*, *regrid()*)
- mechanisms to create components and to control a components lifecycle (*setservices*, *Init()*, *Run()*, *Final()* ...)



Graph of hierarchy in hypothetical ESMF application in left lower panel. Numbers correspond to numbers on left

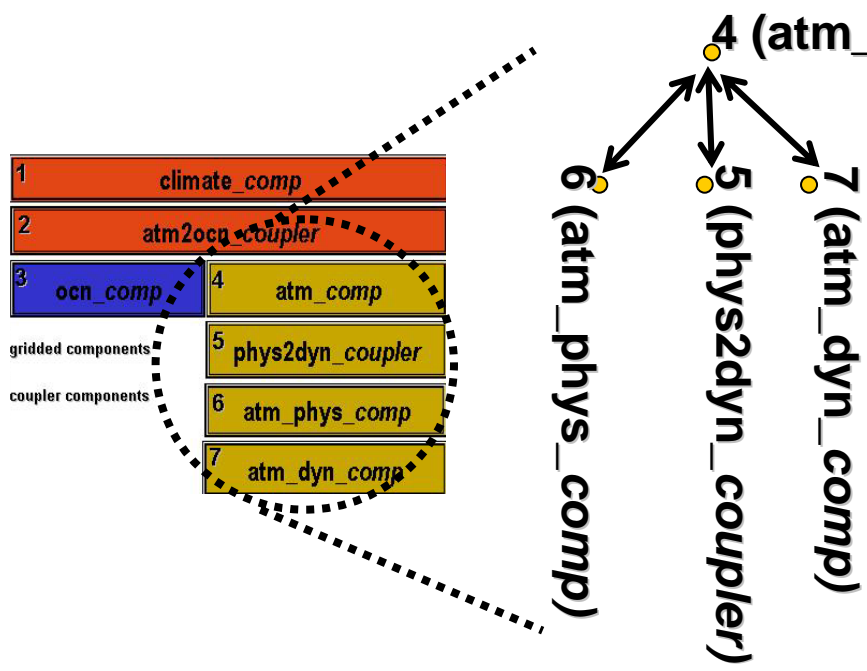
Components are nodes. Flow of data between components is shown by edges.

Example: hypothetical ESMF application



The ESMF Component Model. Wiring I: ESMF_State

- ESMF_state handles data flows **between** components.
- Component model defines a special data type, ESMF_state for these transfers.
- Components exchange data (with their parent in the graph) by encoding/decoding ESMF_state data type arguments.
- An ESMF_state holds different sorts of information, includes meta-data needed to decode it.

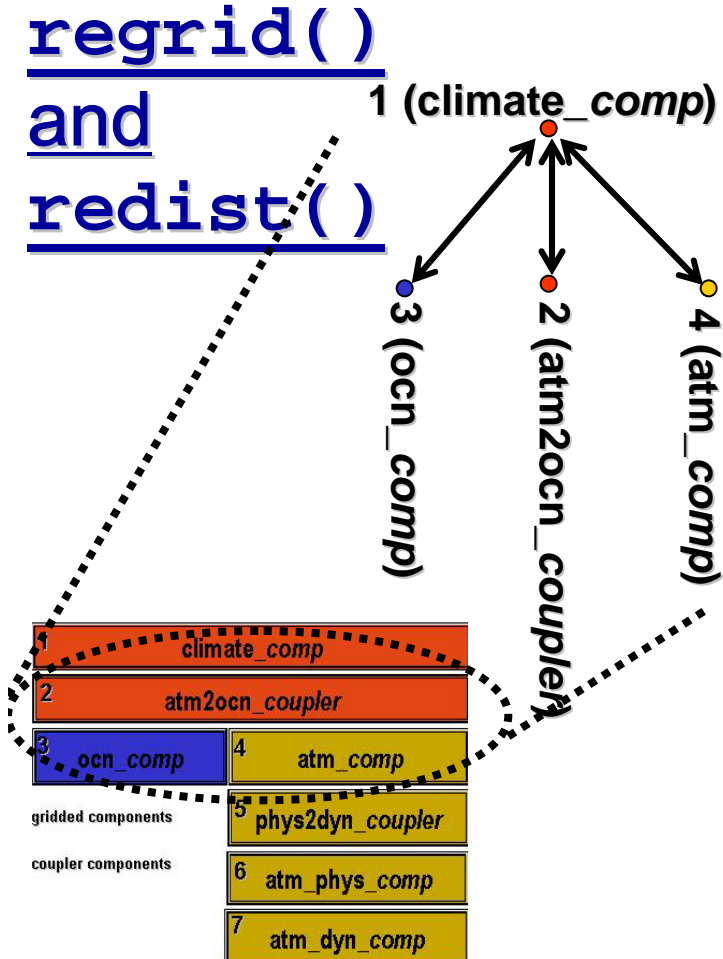


- Typical sequence from highlighted fragment
1. atm_comp calls atm_phys_comp. Inputs for atm_phys_comp encoded in ESMF_State.
 2. atm_phys_comp results encoded in another ESMF_State.
 3. atm_comp calls phy2dyn_coupler using results in 2 as input
 4. phy2dyn_coupler returns its results, another ESMF_State, to atm_comp etc...

The ESMF Component Model Wiring II:

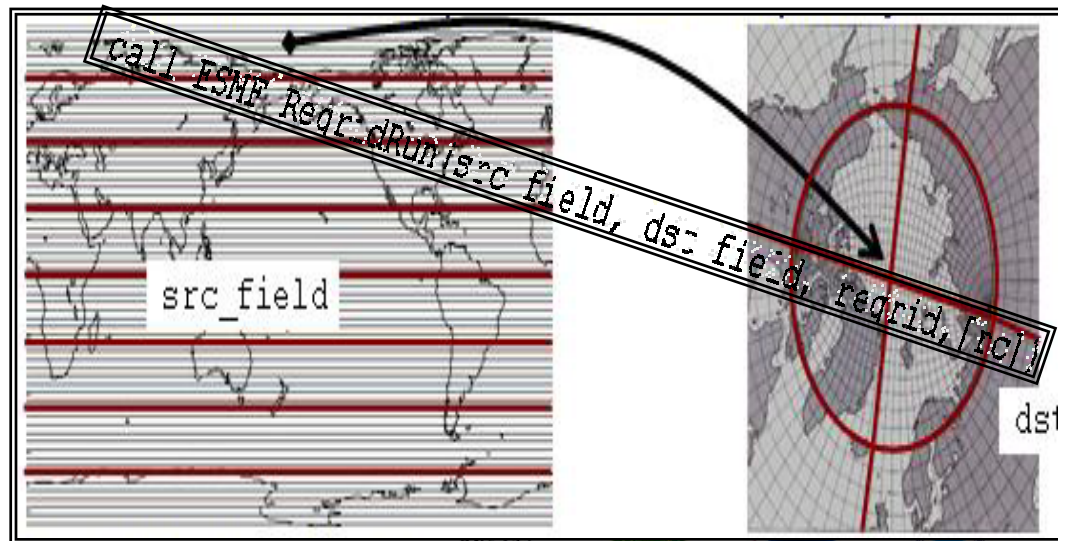
regrid() and redist()

- regrid/redist can manipulate ESMF_state contents – particularly ESMF physical fields and arrays.
- Use regrid/redist to map **within** a component. i.e. ESMF_state still used for “wiring” **between** components.



Typical sequence in highlighted fragment

1. climate_comp gets ESMF_state with left grid + decomposition below from atm_comp.
2. climate_comp passes ESMF_state to atm2ocn_coupler
3. atm2ocn_coupler maps to ocean_comp grid + decomposition (on right below). Returns regridded field within ESMF_state



The ESMF Component Model.

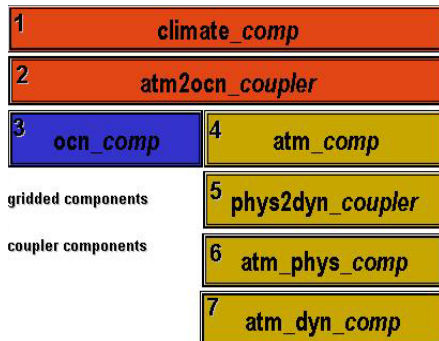
Creating and controlling components

component model defines a standard set of interfaces that a component must support.

Key ones `Initialize(IState,EState,Clock,RC)`,
`Run(IState,EState,...)` `Finalize(IState,EState,...)`

Typical component life cycle is

1. Parent creates a component variable
`child=GridCompCreate()`
2. Parent gets component to initialize its interface list
`CALL GridCompSetServices(child,childSS)`
3. Parent initializes and then executes the component (usually repeatedly)
`CALL GridCompInitialize(child,imp,exp,clk,rc)`
`CALL GridCompRun(child,imp,exp,clk,rc)`



A component also carries private internal state pointers. These enable “child” to remember private state between calls.

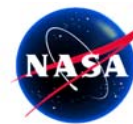


The ESMF Component Model. Synopsis

- Data flows from component to component hierarchically.
- An ESMF application makes a tree is an acyclic graph.
- Data is transferred **between** components through **ESMF_state** type “container variables”.
- Functions such as **regrid/redist** are used in coupler components to succinctly remap data **within** components.
- Every component registers **Init()**, **Run()**, **Finalize()** so that it can be controlled from a parent.
- Components may use an internal state pointer to maintain private state between calls.
- The component model provides everyone with a common model of time and alarms at the component interface.

These aspects of the ESMF component model are being used both to develop interoperable applications and to develop new applications.

So far no AD has been applied.



Talk Outline

ESMF

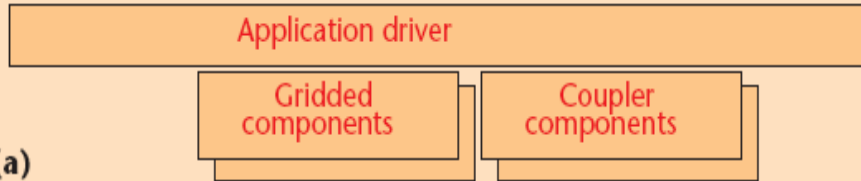
- ESMF Goals
- The ESMF Component Model
 - Component hierarchies
 - Dataflow between components
 - Controlling components
- Under the hood
- AD on this sort of large-scale, multi-component application



Visualizing the component architecture



1. ESMF provides an environment for assembling components



(a)

2. ESMF provides a toolkit that components use to

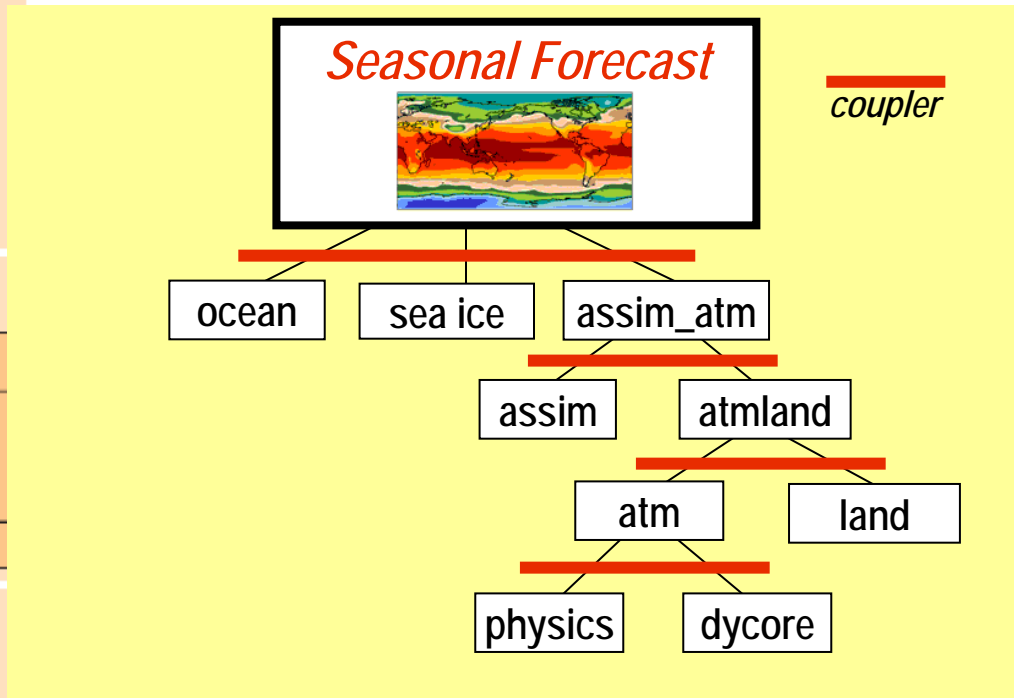
- ensure interoperability
- abstract common services

Component run(), checkpoint():	
Field: halo(), import(), export() + I/O	Grid: regrid(), transpose() + Metrics
DELayout, PEList, Machine model	

(b)

3. Gridded components and coupler components are user written

(c)



Creating a component

```
TYPE(ESMF_GridComp) :: COMPONENT1
```

Sets up a table of function pointers

Phase 1. *Creation* creates components and registers user-code procedures for methods called in later phases. Create steps also are called for gridded component COMPONENT2 and coupler component COUPLER21 (not shown).

```

:
COMPONENT1 = EMSF_GridCompCreate("Example Component 1", DELAYOUT_GC1)
CALL ESMF_GridCompSetServices (COMPONENT1,component1_register)
COUPLER12  = EMSF_CplCompCreate("Example Coupler12", DELAYOUT_CPL12)
:
    
```

Figure 4. Simplified Fortran-like pseudo code for two gridded components. COMPONENT1 and COMPONENT2 communicate with one another through two coupler components, COUPLER12 and COUPLER21.

Initializing a component

Calls one of the functions registered in the function pointer table. Function has standard “container” args IMPORT1, EXPORT1.

Phase 2. *Initialization* calls the user-code initialization procedures registered in phase 1. Initialize steps are also called for COMPONENT2, COUPLER12, COUPLER21 (not shown)

```
:  
CALL ESMF_GridCompInitialize(COMPONENT1, IMPORT1, EXPORT1, CLOCK, RC)
```

Figure 4. Simplified Fortran-like pseudo code for two gridded components. COMPONENT1 and COMPONENT2 communicate with one another through two coupler components, COUPLER12 and COUPLER21.

IMPORT1, EXPORT1 are linked lists in which contain pointers to a number of ESMF types for holding actual data.



Running a component

Phase 3. *Run* calls the user code's run procedures, normally within one or more control loops (not shown). At each loop iteration, gridded component `COMPONENT1` receives import and export ESMF state objects, `IMPORT1` and `EXPORT1`. The run procedure of coupler component `COUPLER12` maps between `EXPORT1`, the export ESMF state object of `COMPONENT1`, and `IMPORT2`, the import ESMF state object of gridded component `COMPONENT2`. The coupler component `COUPLER21` acts in the opposite sense. It maps the `COMPONENT2` export ESMF state, `EXPORT2`, onto `IMPORT1`, the import ESMF state object of `COMPONENT1`. The run procedures use values set (not shown) in the ESMF clock object `CLOCK`.

:

```
CALL ESMF_GridCompRun(COMPONENT1, IMPORT1, EXPORT1, CLOCK, RC)
CALL ESMF_CplCompRun(COUPLER12, EXPORT1, IMPORT2, CLOCK, RC)
CALL ESMF_GridCompRun(COMPONENT2, IMPORT2, EXPORT2, CLOCK, RC)
CALL ESMF_CplCompRun(COUPLER21, EXPORT2, IMPORT1, CLOCK, RC)
```

[putting it all together](#)



Talk Outline

ESMF

- ESMF Goals
- The ESMF Component Model
 - Component hierarchies
 - Dataflow between components
 - Controlling components
- Under the hood
- AD challenges on this sort of large-scale, multi-component application



“Analyze this” – AD challenges for component arch.



From user perspective

Clean abstraction and modularity for person wiring together an application.

From AD perspective

Cross-component data flows and dependency are buried in specialized types.

Cross-component data flows are almost always linear steps (not dependent on local state) e.g. permute, average..

Q - What is the best approach to expose information to an AD tool?

