The use of modeling frameworks to facilitate interoperability

Cecelia DeLuca/NCAR (ESMF)
Bill Putman/NASA GSFC (MAPL)
David Neckels/NCAR (ESMF)

June 4, 2008
ASP Colloquium

ESMF

Outline

1st half…

- What is a framework and why is it useful?
- Some background on ESMF
- Parallel regridding in ESMF

2nd half…

- Using MAPL to wrap components

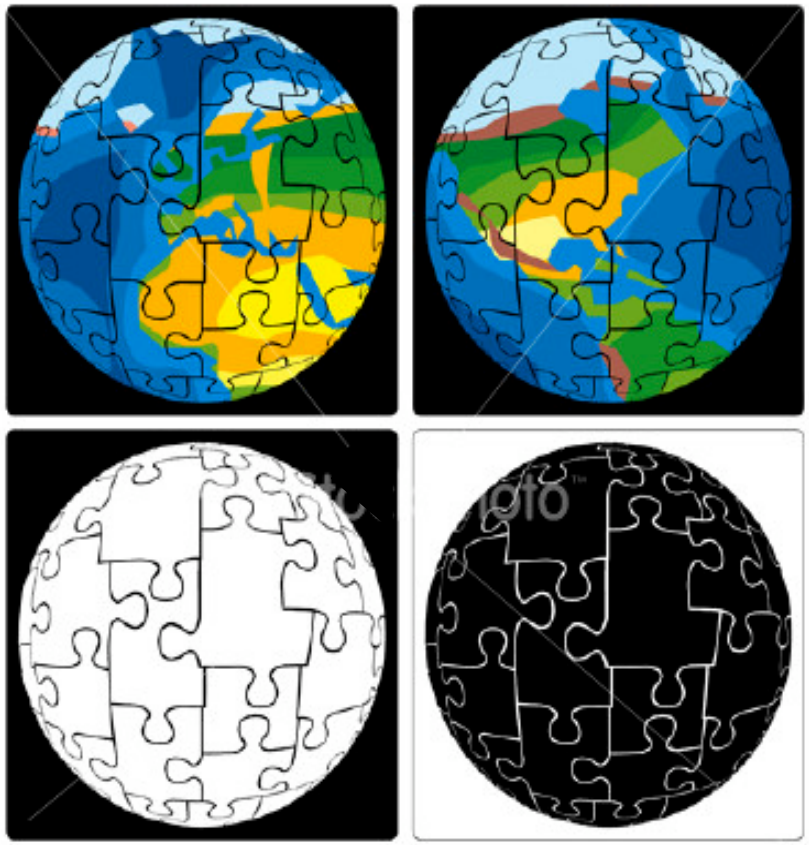
What is a framework?

In computer science

- "Frameworks" are partially completed software applications that users customize and finish using elements of their own codes
- The prefabricated pieces of the framework add capabilities and structure to the user code

In the Earth sciences

- The term "framework" is loosely applied to many software packages useful for building models



Benefits of using frameworks

Efficiency of development

- Modelers can save development and maintenance effort by leveraging toolkits for common functions, such as calendar management and message logging

Access to numerical and computational advances

- Frameworks can make advanced numerical techniques, software for optimized operations on petascale computers, and other specialized capabilities accessible to a broad set of users

Interoperability

- When many groups use the same or similar frameworks, it is simpler to exchange and combine model components (such as dynamical cores)
- The comparison, evaluation, and validation of individual model components is easier if components are cleanly separable and can participate in controlled experiments

Frameworks in Earth and space Sciences

Many different efforts, distinguished by:

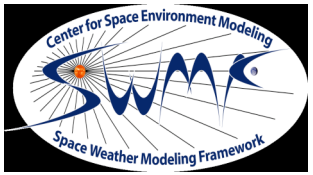
Institution - e.g. Flexible Modeling System (FMS) at GFDL

Domain - e.g. Space Weather Modeling Framework

Scope - e.g. cross-domain Common Component Architecture

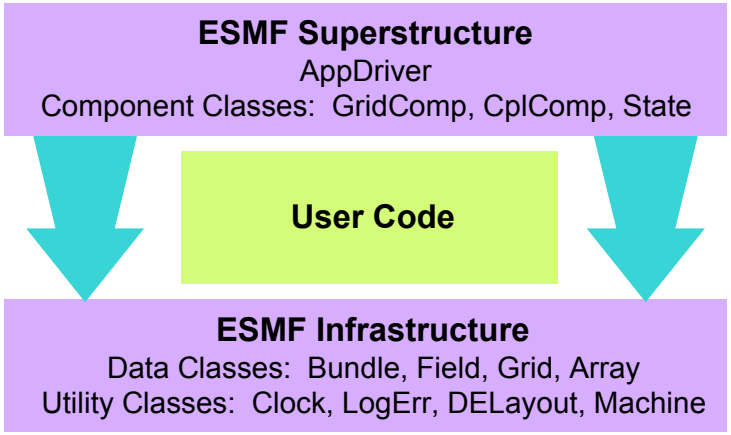
Computing Platform - e.g. OpenMI for Windows

Technical Strategy - e.g. PRISM climate framework for coupling components that run as separate programs



Earth System Modeling Framework (ESMF)

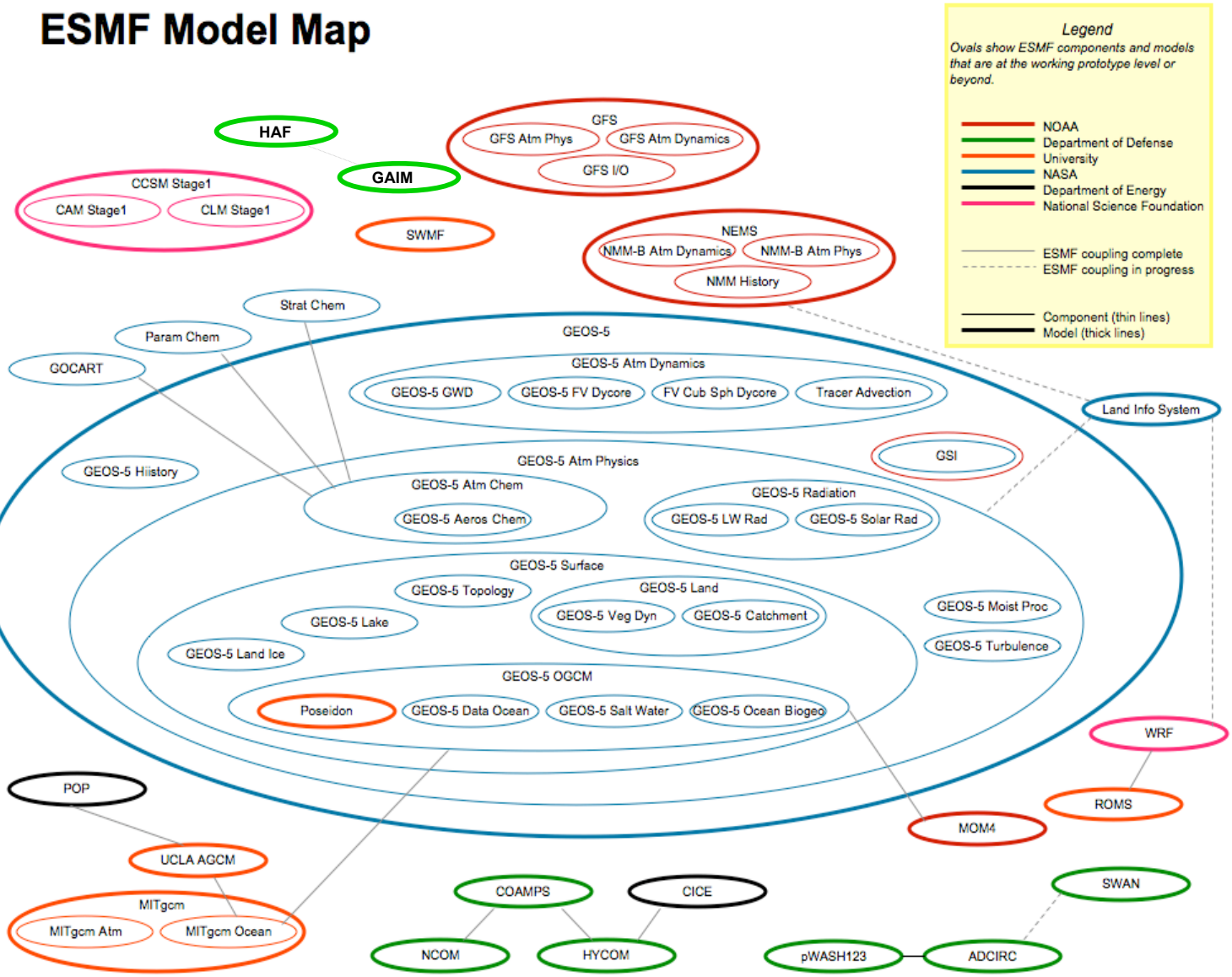
- ESMF provides standard interfaces for model components
- ESMF provides common utilities and tools for routine modeling functions such as regridding between components



Status

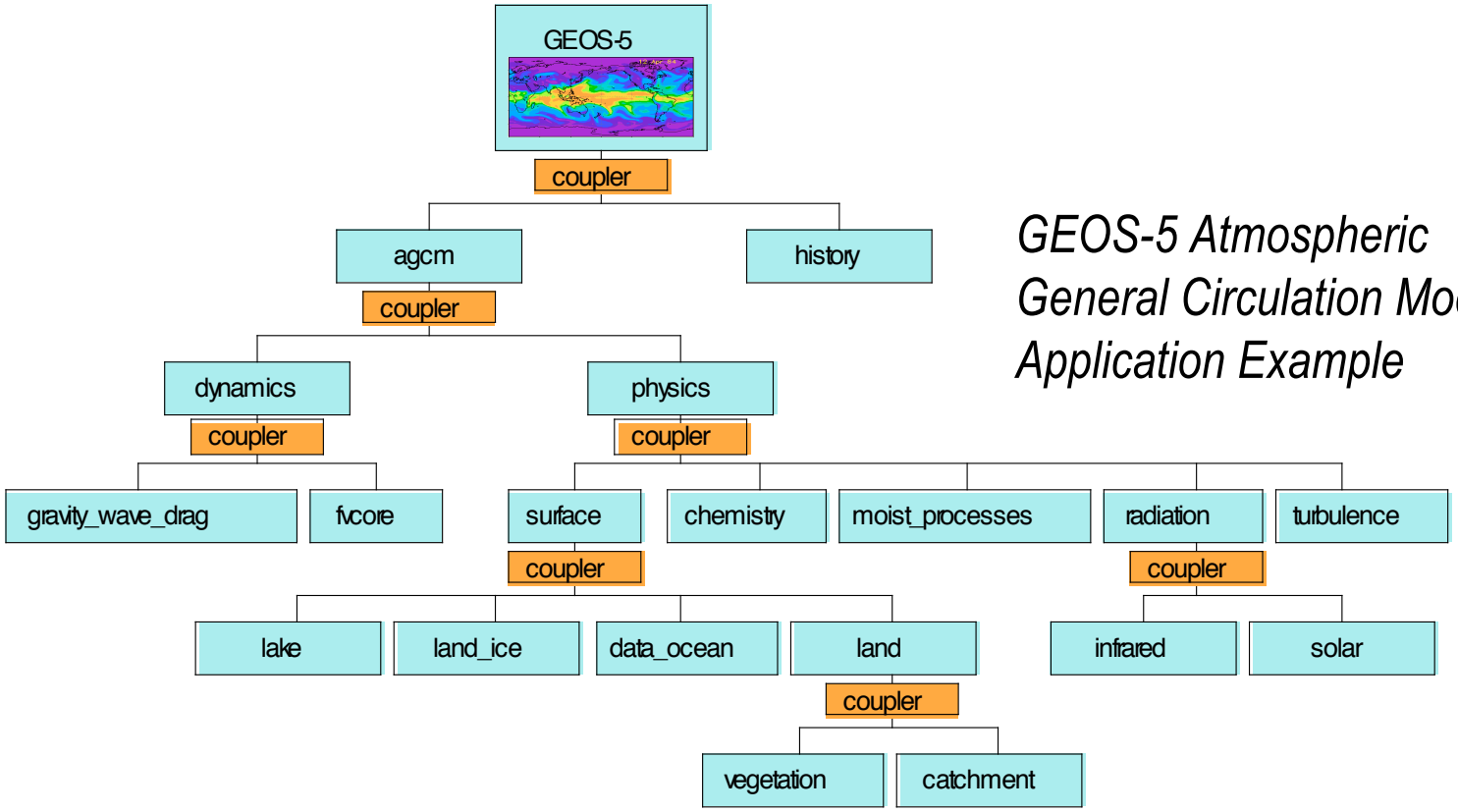
- Initiated by NASA in 2002 and developed and managed by a multi-agency consortium
- Used for coupling climate, weather, hydrological, biological, space weather, and other components (including 3 dynamical cores in this workshop) running on high performance computing platforms
- More than 60 ESMF components in the community
- Highly portable and scalable

ESMF Earth Science Components & Models



ESMF

ESMF Application Example

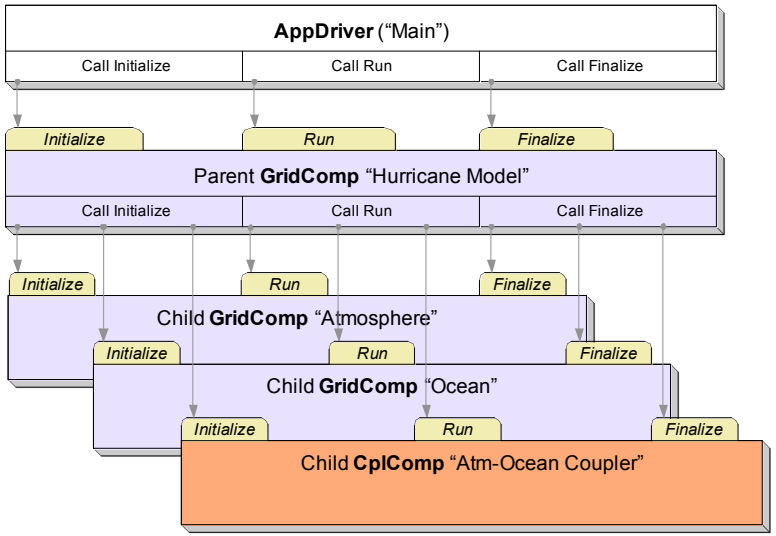


GEOS-5 Atmospheric General Circulation Model Application Example

- Each box is an ESMF gridded component or coupler component
- ESMF State objects carry data between components
- Every component (including couplers) has a standard interface to facilitate exchanges
- The ESMF architecture enables the assembly of many different systems

Interoperability and standardization of interfaces

- Standard component interfaces are the basis for modularity and interoperability
- There are only three ESMF component methods: Initialize, Run, and Finalize (I/R/F)
- Users assign their user code I/R/F methods to an ESMF Component type, thereby "finishing" the Component
- The ESMF Component calls back into the specific user-assigned methods
- I/R/F methods cascade down the tree
- Small set of standard arguments:



call ESMF_CompRun (myComp, importState, exportState, clock, phase, blockingFlag, rc)

ESMF Adoption – PARSE

Prepare user code

- Decide on components, coupling fields and control flow and split user code into I/R/F methods

Adapt data structures

- Wrap user data structures either as
  1. ESMF Arrays – represent user data in index space
     Regrid via user-supplied interpolation weights input to ESMF sparse matrix multiply
  2. ESMF Field objects – include coordinates and so represent user data in physical space
     Regrid using ESMF parallel interpolation weight generation in v3.1.1

  Pack Arrays or Fields into States.
  Wrap time information in ESMF Clocks.

Register user methods

- Attach user code I/R/F methods to ESMF Components by calling registration methods

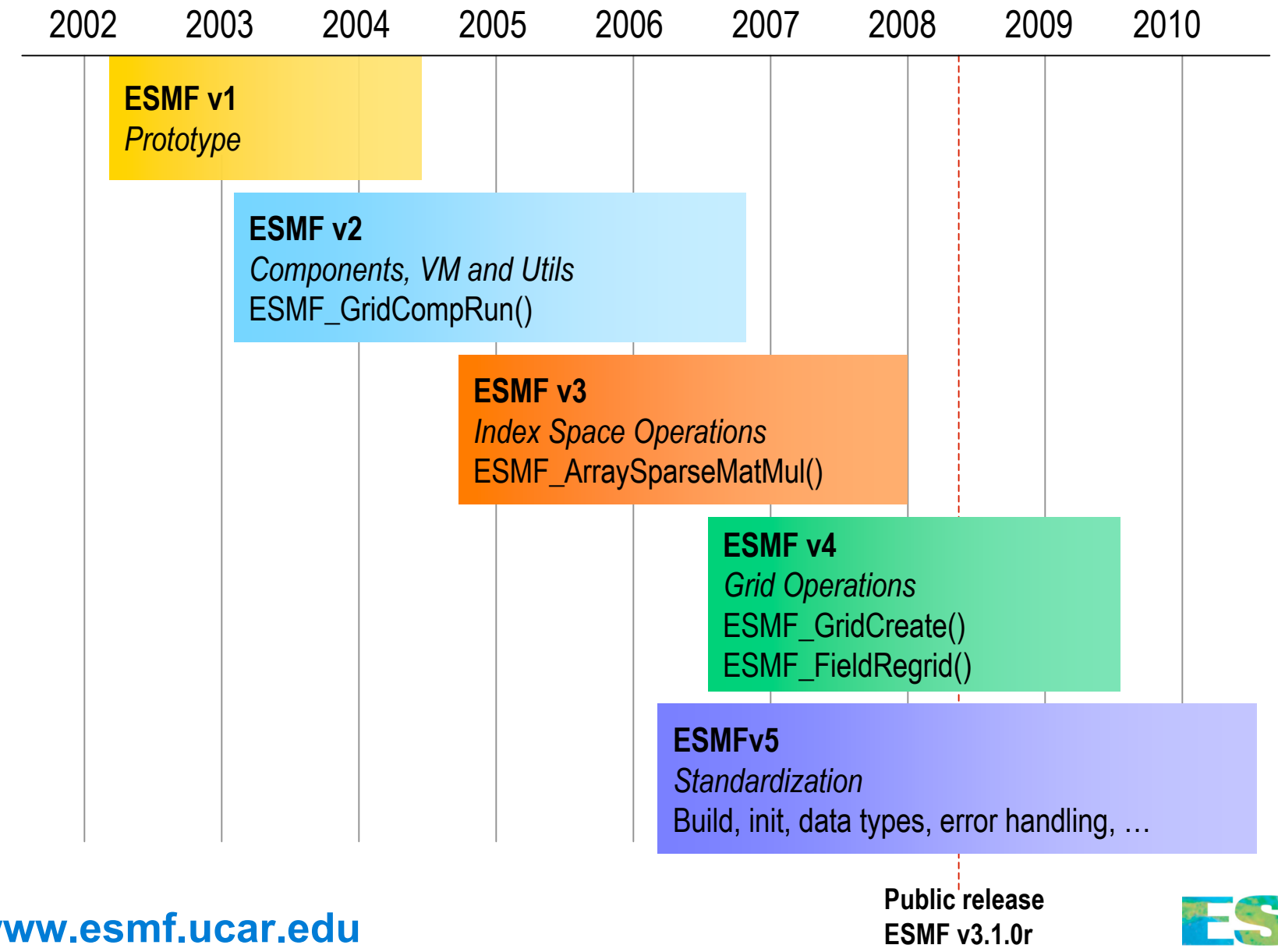
Schedule, synchronize, and send data between components

- Write couplers using ESMF redistribution, sparse matrix multiply, regridding, or user-specified transformations

Execute the application

- Run components using an ESMF driver

Release Path



Public Release v3.1.0r, May 2008

- Open source, open development, on-line browsable repository
- Serial (one processor) or parallel (many processors)
- Components can run concurrently (components run on mutually exclusive processors), sequentially (all components run on the same processors) or in mixed mode
- Single executable (all components run as one big program) or multiple executable (components run as separate programs) or combinations
- Shared or distributed memory or hybrid
- Support for model ensembles, including execution of multiple ensemble members in the same address space
- 2000+ unit tests, system tests, and examples regression tested nightly on 26+ platform/compiler combinations
- Exhaustive Reference Manual and Users Guide

More on the current release …

- Data transformations can be executed within a coupler component, or arranged in a coupler component and executed directly between model components
- Coupling can be done in index space or physical space
- Performance: < 5% overhead in time to solution vs customized native approaches, highly scalable in performance and memory (performance reports online)

Upcoming Release

Internal Release 3.1.1, June/July 2008

- Representation of observational data streams
- Generation of regridding interpolation weights for logically rectangular grids, bilinear and higher order methods (conservative coming)
- Attribute class can store and write standard metadata packets and represent metadata hierarchies - e.g., State metadata includes the metadata of Fields that are stored in it

The last item connects ESMF to Curator, since it enables ESMF metadata output to be input into a web portal that describes components, models, and experiments – and it is also a first step to automating coupling

How to Get Started & Get Help

- Support list – esmf_support@ucar.edu
- Web meeting with the development team
- Code Examples webpage and FAQ
- Tutorials and Coding Workshops
- Reference Manual and Users Guide

Parallel Rendezvous Regridding in ESMF

David Neckels and Cecelia Deluca

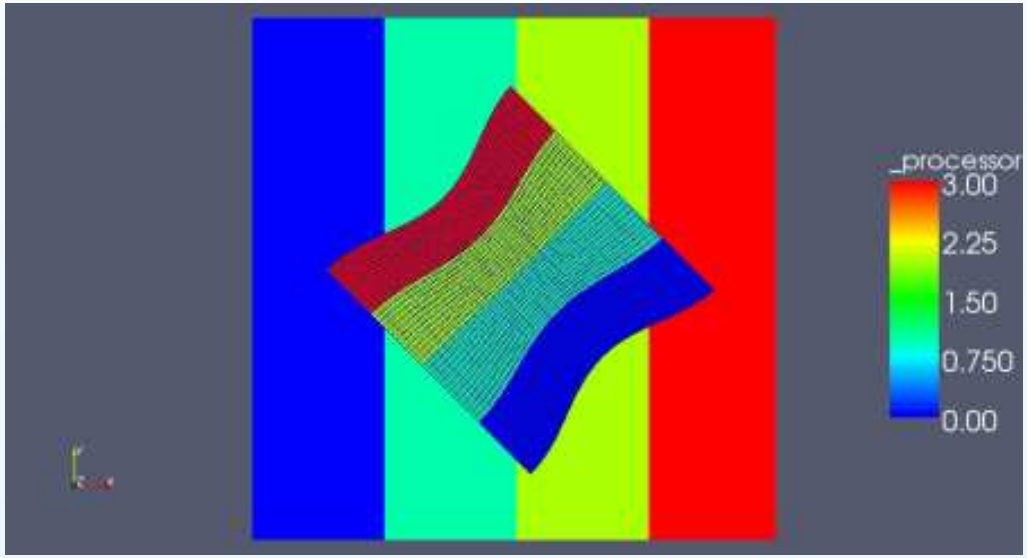
National Center for

Atmospheric Research

Parallel Rendezvous

Example ESMF grids

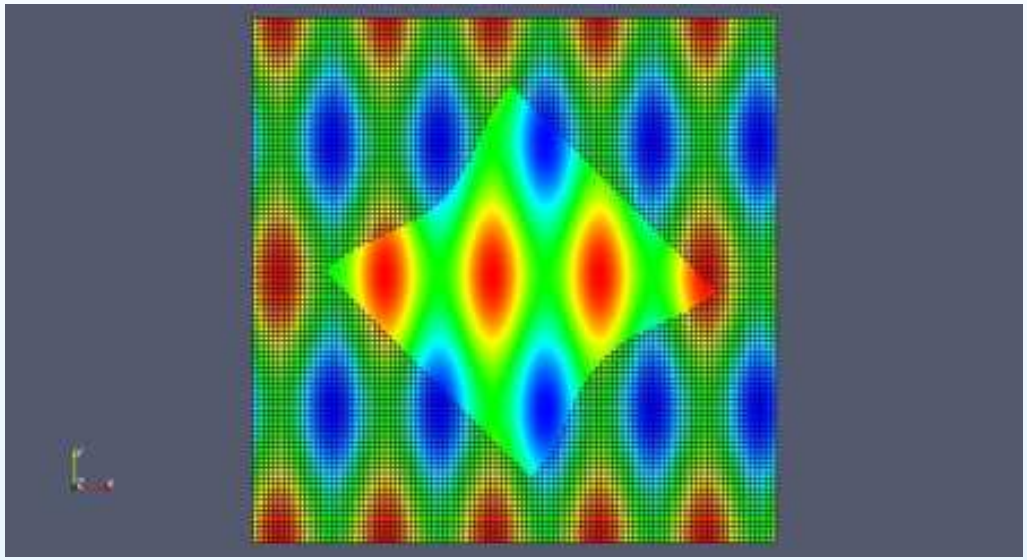
ESMF computes a regridding operator from source to destination grid:



With petascale applications and very large grids in mind, we compute this operator in parallel.

Example ESMF grids

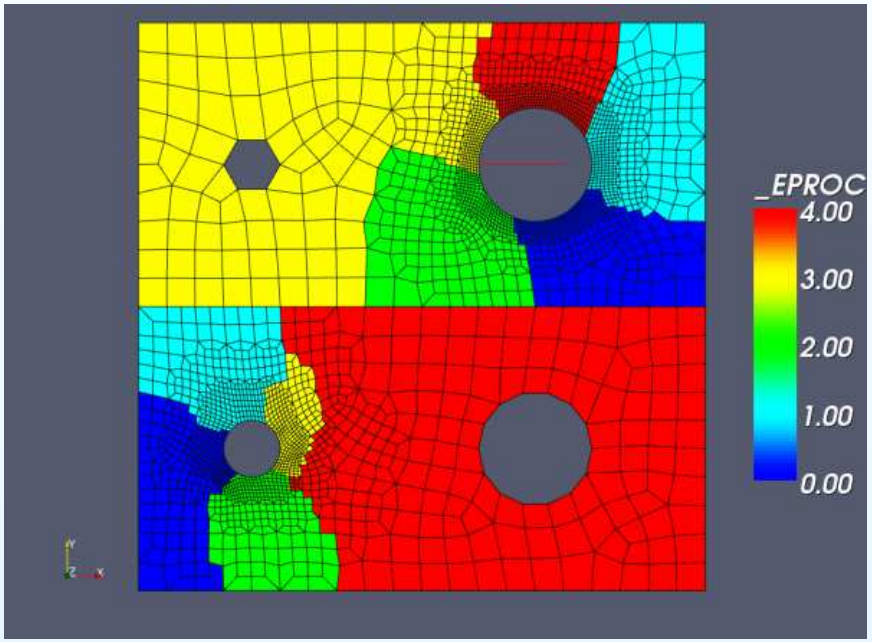
ESMF also implements the regridding operator, which operates on ESMF Array objects



to transfer field values to the destination grid, using a highly optimized sparse matrix multiply kernel.

Example ESMF grids

More generally, ESMF can build regridding operators for mesh <-> mesh and mesh <-> grid interactions



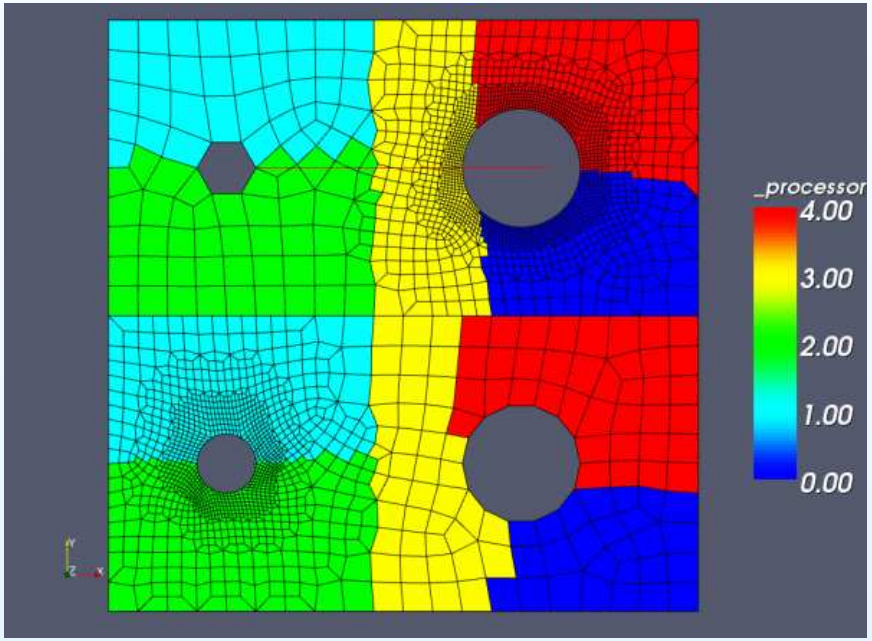
These grids/meshes will not likely be geometrically aligned.

Geometric Rendezvous

We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!

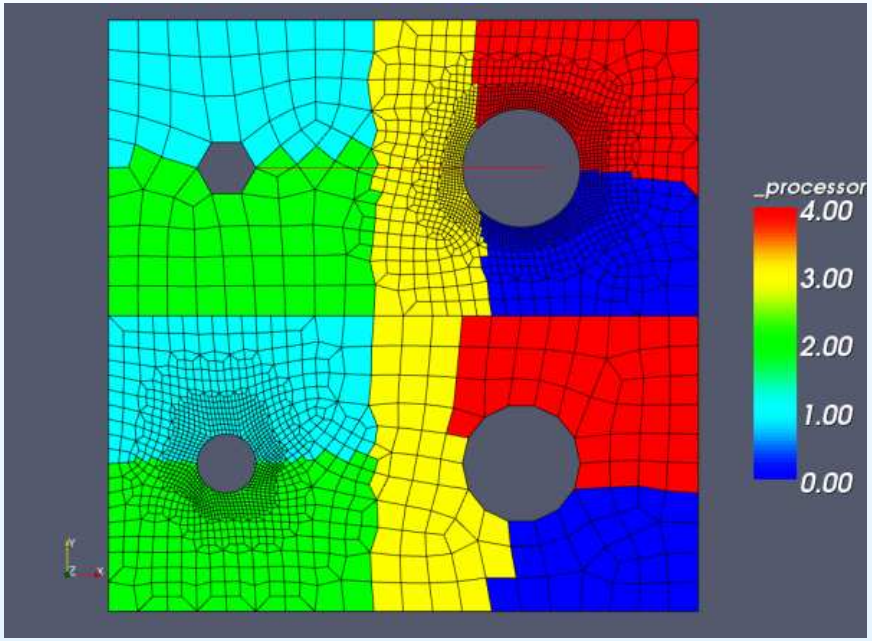
Geometric Rendezvous

We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!



Geometric Rendezvous

We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!



Also, the union of meshes is load balanced, so that the local searches are optimal.

Spherical Interpolation

Global Interpolation

Global interpolation to and from spherical grids is ill defined as a two dimensional problem, as it is usually solved.

Global Interpolation

Global interpolation to and from spherical grids is ill defined as a two dimensional problem, as it is usually solved.

Branch cuts in each grid complicate search algorithms.

Global Interpolation

Global interpolation to and from spherical grids is ill defined as a two dimensional problem, as it is usually solved.

Branch cuts in each grid complicate search algorithms.

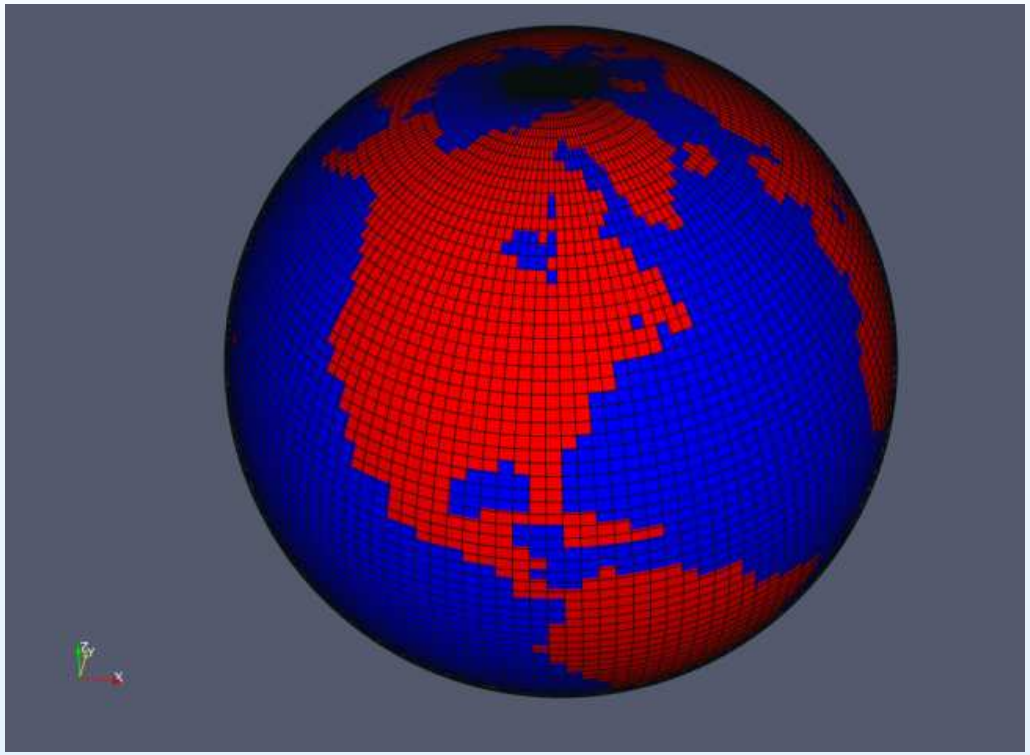
The pole singularity is not handled correctly

Generalization is 3D

The best generalization of the various representations of the sphere?

Generalization is 3D

The best generalization of the various representations of the sphere?



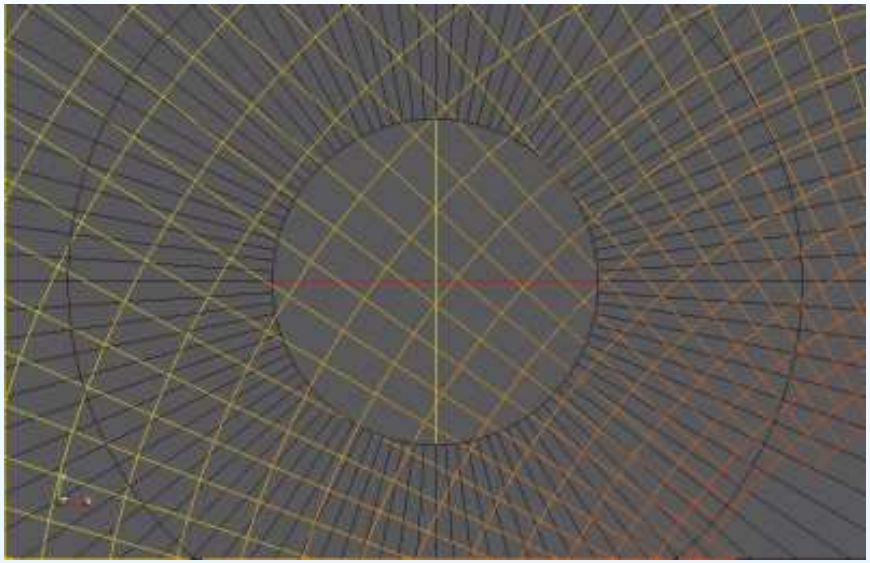
We represent these grids a two dimensional manifold in 3-space.

Pole/Extrapolation issues

We propose unique solutions to pole and extrapolation regrid difficulties

Pole/Extrapolation issues

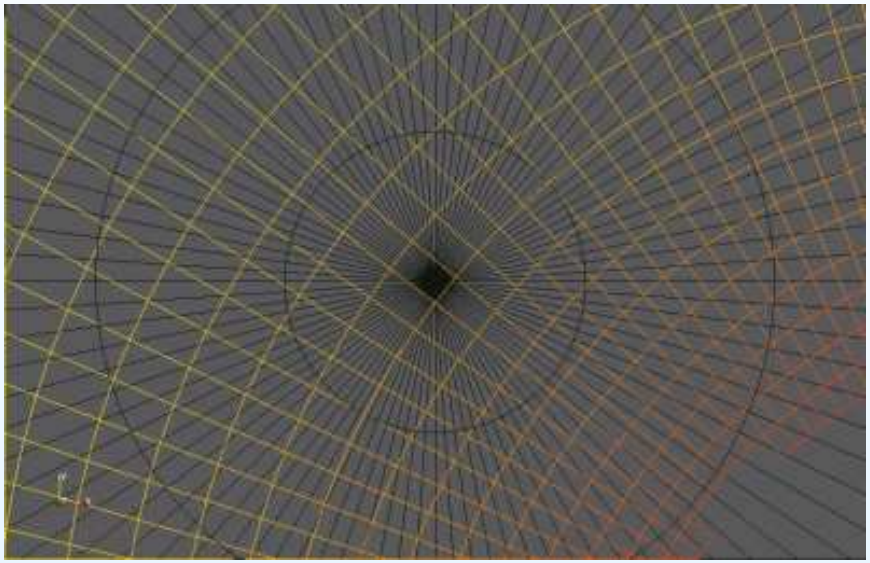
We propose unique solutions to pole and extrapolation regrid difficulties



Example, Holes at the pole (regular lat/lon and POP grids)

Pole/Extrapolation issues

We propose unique solutions to pole and extrapolation regrid difficulties



We triangulate the area and introduce a constrained degree of freedom at the pole.

Pole/Extrapolation issues...

Given this constrained degree of freedom, we form the interpolation matrix $\tilde{M}$ in the usual way (e.g. bi-linear/tri-linear interpolation).

Pole/Extrapolation issues...

Given this constrained degree of freedom, we form the interpolation matrix $\tilde{M}$ in the usual way (e.g. bi-linear/tri-linear interpolation).

We form the constraint matrix C, which describes how field values at the pole (or other constrained nodes) are to be manufactured from the true degrees of freedom.

Pole/Extrapolation issues...

Given this constrained degree of freedom, we form the interpolation matrix $\tilde{M}$ in the usual way (e.g. bi-linear/tri-linear interpolation).

We form the constraint matrix C, which describes how field values at the pole (or other constrained nodes) are to be manufactured from the true degrees of freedom.

We finally reduce out the constraints, forming the final interpolation matrix $M = \tilde{M}C$.

Pole/Extrapolation issues...

Given this constrained degree of freedom, we form the interpolation matrix $\tilde{M}$ in the usual way (e.g. bi-linear/tri-linear interpolation).

We form the constraint matrix C, which describes how field values at the pole (or other constrained nodes) are to be manufactured from the true degrees of freedom.

We finally reduce out the constraints, forming the final interpolation matrix $M = \tilde{M}C$.

This approach can also work on grid boundaries, where data may not be available for all cell nodes; one only need provide a suitable extrapolation constraint matrix, C.

Patch Recovery Interpolation

Patch based interpolation

ESMF provides higher order smoothing interpolation.

The Patch method uses local least squares fits to form a smoothing interpolation operator.

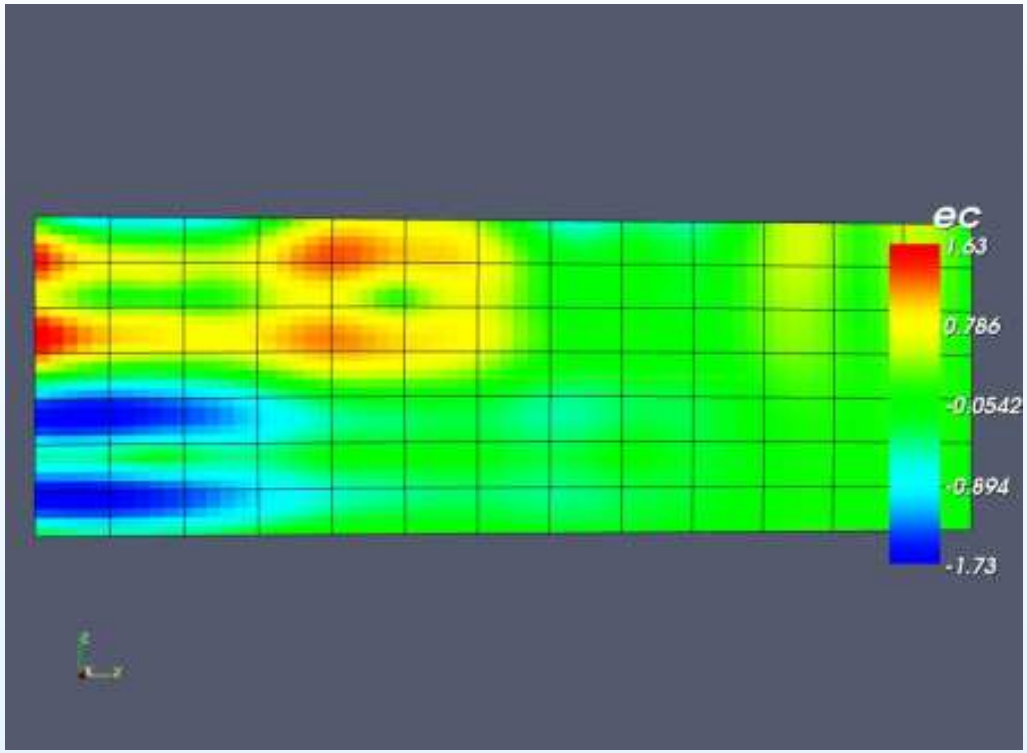
This method is more convenient than bi-cubic, since derivatives are not needed.

Patch based interpolation...

Derivatives of the interpolant from coarse to fine scale grids are greatly improved

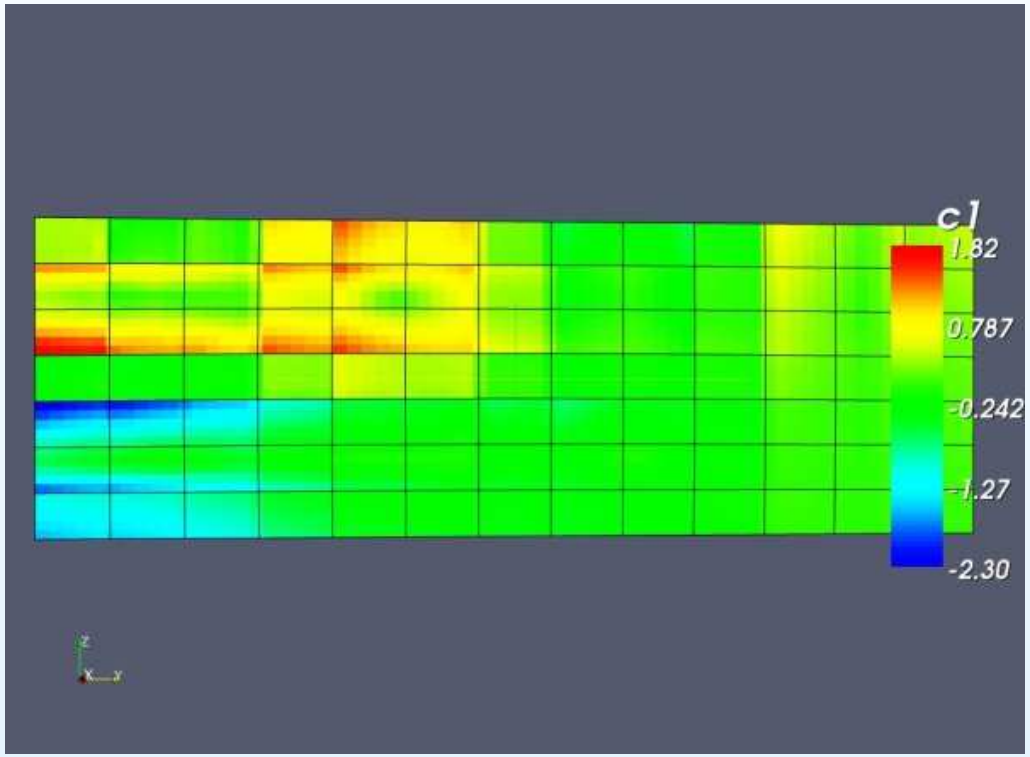
Patch based interpolation...

Analytic curl of a wind stress field on the fine grid



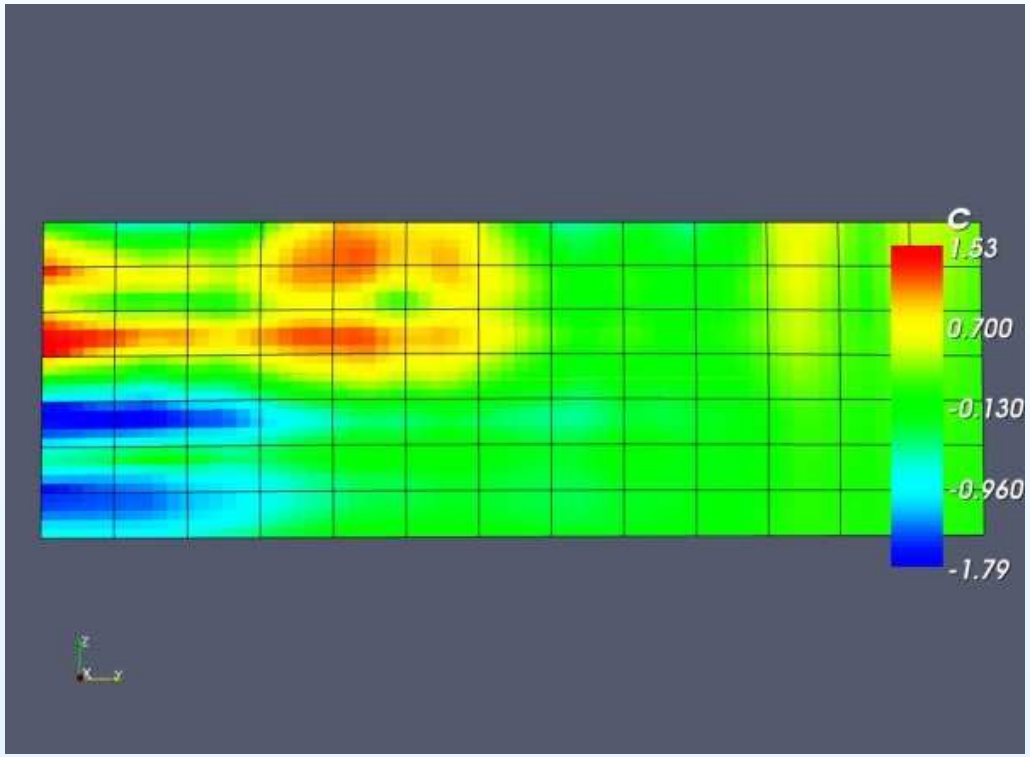
Patch based interpolation...

Curl of a the bilinear interpolant of wind stress field on the fine grid



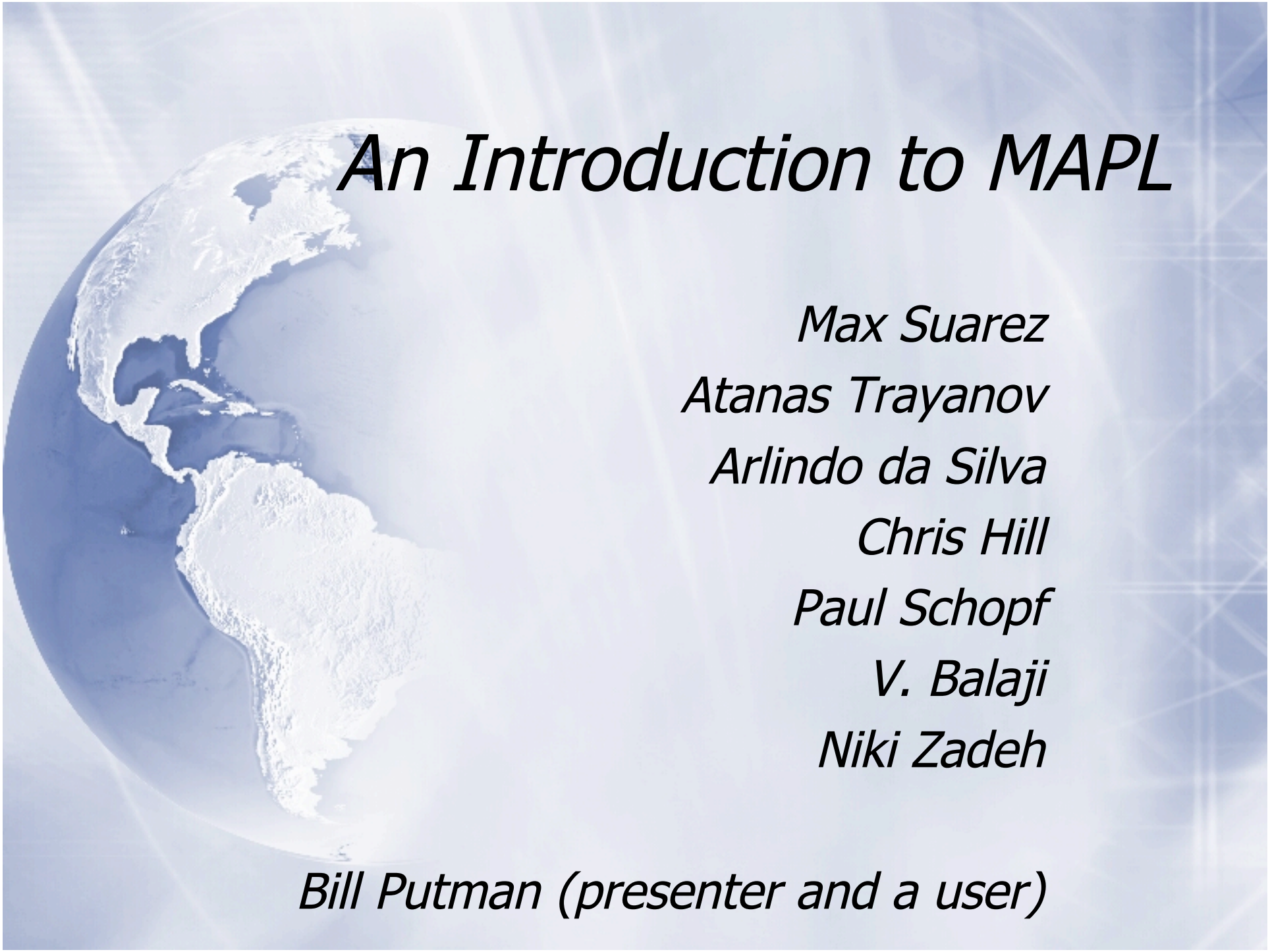
Patch based interpolation...

Curl of a the patch-recovery interpolant of wind stress field on the fine grid



References

- "A parallel rendezvous algorithm for interpolation between multiple grids," Steve Plimpton, Bruce Henderickson, James Stewart. Proceedings of the 1998 ACM/IEEE conference on Supercomputing, 1998.
- "SIERRA Framework Version 3: Core Services Theory and Design," H. Carter Edwards. Sandia National Laboratories, report SAND2002-2616, 2002.
- "Architecture of the Earth System Modeling Framework," Hill, C., C. DeLuca, V. Balaji, M. Suarez, and A. da Silva. Computing in Science and Engineering, Volume 6, Number 1 (2004).
- "Zoltan: Data Management Services for Parallel Dynamic Applications," Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson and Courtenay Vaughan. Computing in Science and Engineering, Vol 4 Number 2, 2002.



An Introduction to MAPL

Max Suarez

Atanas Trayanov

Arlindo da Silva

Chris Hill

Paul Schopf

V. Balaji

Niki Zadeh

Bill Putman (presenter and a user)

Outline

- MAPL Intro / Development History
- Relevant aspects of ESMF
- Role of MAPL (as a middleware layer)
- Core Elements of MAPL
- Basic Recipe for a MAPL Component
- The DynCore_GridComp example

MAPL Introduction

- Objectives
  - Establish standards and software tools for building ESMF compliant Components
  - Facilitate the porting of existing codes to ESMF
  - Provide tools and a clean recipe for building ESMF components
  - Facilitate interoperability between ESMF compliant components

MAPL Introduction

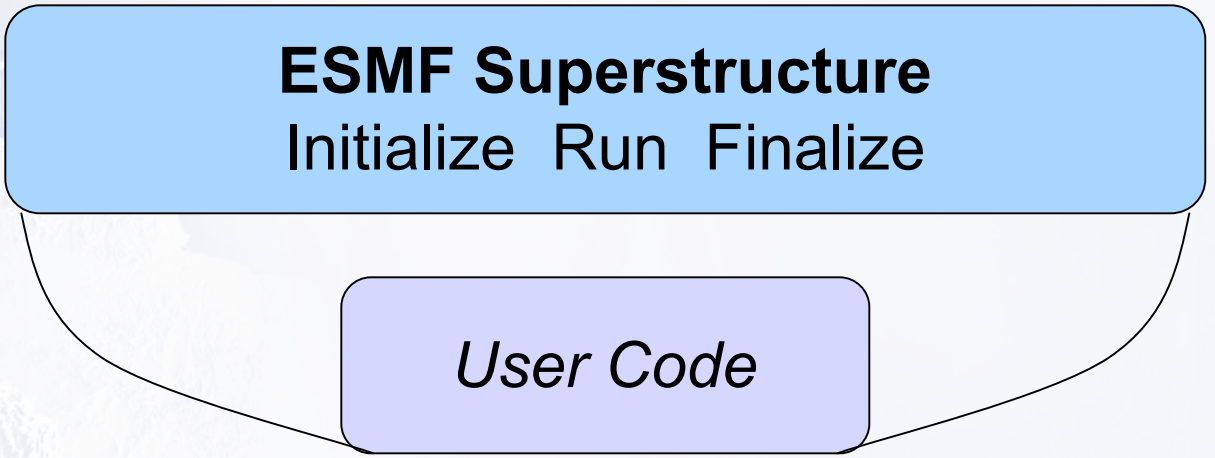
- Development History
  - Arose from the development of GEOS-5 at NASA GMAO
  - Observed that the implementation of ESMF components could be generalized and re-used
  - Intended to be a generic layer to facilitate building a hierarchical structure of ESMF components for Earth System Models

Relevant Aspects of ESMF

- ESMF provides essential functions required by parallel, scalable earth system models in a machine independent way
- ESMF provides general programming classes to construct ESMF components (infrastructure) or connect components to one another (superstructure)

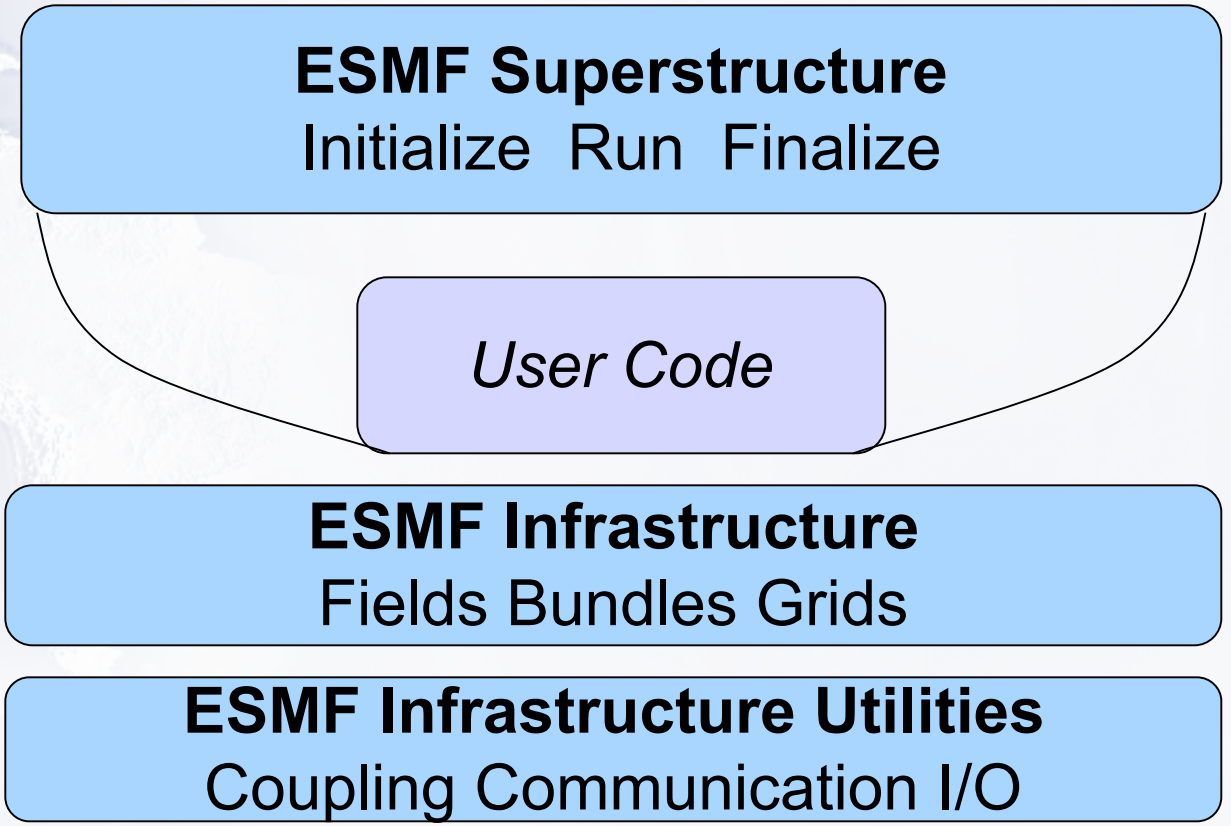
Relevant Aspects of ESMF

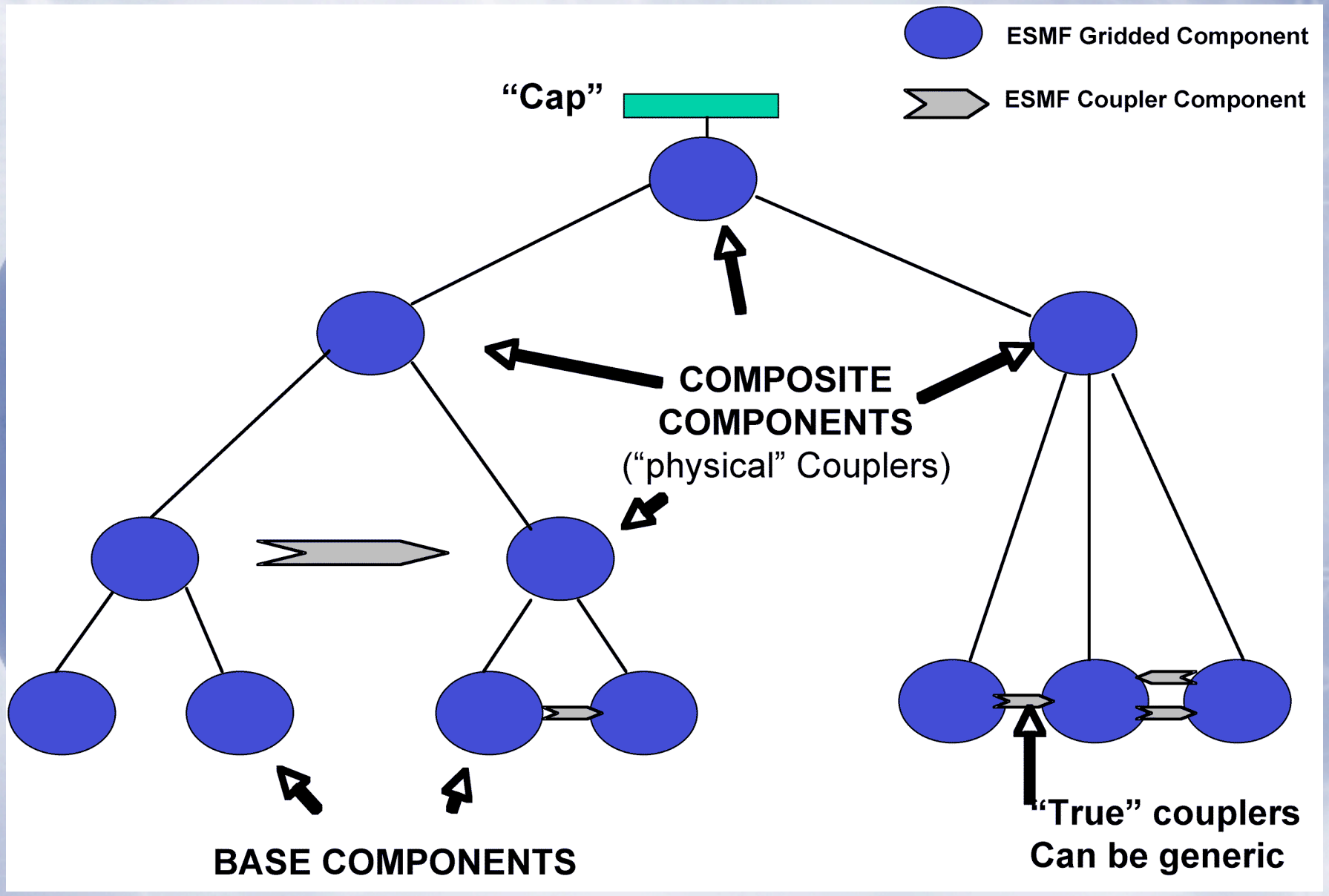
- The simplest implementation



Relevant Aspects of ESMF

- More advanced implementations




ESMF Gridded Component
ESMF Coupler Component
"Cap"
COMPOSITE
COMPONENTS
("physical" Couplers)
BASE COMPONENTS
"True" couplers
Can be generic

MAPL Objectives

- Specify conventions and best practices for ESMF utilization in Climate/Weather models
- Serve as a middleware layer (between user code and ESMF) to facilitate adoption of ESMF in Climate/Weather models

Core Aspects of MAPL

- MAPL_Core
  - Register Init/Run/Finalize Methods
  - Provide Data Services
- MAPL_Connect
  - Define data connections
  - Manage connections among child components (coupling)
  - Perform regridding as needed
- MAPL_History
  - Manage output streams from a MAPL hierarchy
  - Write any fields requested from the Export states
  - Interpolate/average as requested
- MAPL_CFIO
  - Read/write ESMF states/fields
  - Self-describing (netcdf/HDF) and Flat Binary (GrADs)
- MAPL_Utils
  - Profiling, error handling and astronomy

Core Elements of a MAPL Component

- SetServices
  - Allocate the MAPL object
  - Registers component IRF methods
  - Creates and allocates the IM/IN/EX states
  - Define profiling timers
  - Set children's services
- Initialize
  - Read configurations (parameters required for your run)
  - Query the grid (modify for your component as needed)
  - Initialize children
- Run
  - Advance the component and it's children
- Finalize
  - Checkpoint IM/IN states as requested
  - Clean up IM/IN/EX states

MAPL_Core

- Provide tools for describing a components import/export states
- Internal State is added to extend the ESMF state concept for persistent component data
- Facilitate the use of ESMF Fields and thus of the ESMF Infrastructure layer

```
call MAPL_AddImportSpec(STATE, &
                        SHORT_NAME = 'PLE', &
                        LONG_NAME = 'air_pressure', &
                        UNITS = 'Pa', &
                        DIMS = MAPL_DimsHorzVert, &
                        VLOCATION = MAPL_VLocationEdge, &
                        RC=STATUS )
```

MAPL_Core

- Assist in constructing Init/Run/Finalize (IRF) methods
- Provide generic IRF routines for simple components

```
call MAPL_GridCompSetEntryPoint ( gc, ESMF_SETINIT,  Initialize, rc=status)
VERIFY_(STATUS)
call MAPL_GridCompSetEntryPoint ( gc, ESMF_SETRUN,   Run, rc=status)
VERIFY_(STATUS)
call MAPL_GridCompSetEntryPoint ( gc, ESMF_SETRUN,   RunAddIncs, rc=status)
VERIFY_(STATUS)
call MAPL_GridCompSetEntryPoint ( gc, ESMF_SETFINAL, Finalize, rc=status)
VERIFY_(STATUS)
call MAPL_GridCompSetEntryPoint ( gc, "ESMF_ReadRestart", Coldstart, rc=status)
VERIFY_(STATUS)
```

MAPL_Connect

- Describes the connectivity between components

```
call MAPL_AddConnectivity ( GC,                              &
     SHORT_NAME  = (/ 'DUDT', 'DVDT', 'DTDT' /),   &
     SRC_ID    = PHS,                                  &
     DST_ID    = DYN,                                  &
                                  RC=STATUS  )
  VERIFY_(STATUS)

  call MAPL_AddConnectivity ( GC,                               &
     SRC_NAME = (/ 'U   ', 'V   ', 'T   ', 'PLE  ' /),       &
     DST_NAME = (/ 'U   ', 'V   ', 'TEMP', 'PLE  ' /),   &
     SRC_ID  = DYN,                                          &
     DST_ID  = PHS,                                          &
                                  RC=STATUS  )
  VERIFY_(STATUS)
```

MAPL_Connect

- Simplifies the creation of hierarchical Components
  - Provides a single main program (the MAPL_CAP component) with 2 children (Root and History)
- Automatically creates ESMF coupler components
- Provides support for regridding through Exchange grids

MAPL_History

- An Internal MAPL Gridded Component
- Configuration defined in HISTORY.rc
  - Defines collections of output streams
  - Specifies format (GrADS or CFIO)
  - Specifies "instantaneous" or "time-averaged" fields
  - Beginning and end times for each collection
  - Specifiy frequency of output collections
  - Describe requested time averaging
  - Request regridding (horizontal or vertical)
  - Request fields to be output

MAPL_History

```
EXPID:  fvcubed
EPDSC: fvcubed_CASE-2-0-1234_high_L26

COLLECTIONS:
    ' fvcubed_CASE-2-0-1234_high_L26 '
    ::

fvcubed_CASE-2-0-1234_high_L26.template:  '%y4%m2%d2_%h2%n2z',
fvcubed_CASE-2-0-1234_high_L26.format:    'CFIO',
fvcubed_CASE-2-0-1234_high_L26.descr:     'fvcubed_dycore'
fvcubed_CASE-2-0-1234_high_L26.frequency:  240000,
fvcubed_CASE-2-0-1234_high_L26.resolution: 720 361
fvcubed_CASE-2-0-1234_high_L26.fields:    'PHIS'    , 'DYNAMICS'       ,
                  'PS'      , 'DYNAMICS'       ,
                  'PL'      , 'DYNAMICS', 'P'    ,
                  'U'       , 'DYNAMICS'       ,
                  'V'       , 'DYNAMICS'       ,
                  'T'       , 'DYNAMICS'       ,
                  'TRACER_1' , 'DYNAMICS', 'Q1'   ,
                  'TRACER_2' , 'DYNAMICS', 'Q2'   ,
                  'TRACER_3' , 'DYNAMICS', 'Q3'   ,
                  'TRACER_4' , 'DYNAMICS', 'Q4'   ,
                  'TRACER_5' , 'DYNAMICS', 'Q5'   ,
                  'TRACER_6' , 'DYNAMICS', 'Q6'   ,
              ::
```

MAPL_CFIO

- Interfaces CFIO to the ESMF data types
- Designed for two modes of I/O
  - Self-Describing (netcdf or HDF)
  - Flat Binary (GrADs readable)
- Contains only four methods
  - MAPL_CFIOCreate
  - MAPL_CFIOWrite
  - MAPL_CFIORead
  - MAPL_CFIODestroy
- Can automatically create ESMF arrays on Read
- Independent of all other aspects of MAPL

MAPL_Utils

- Error Handling

```
VERIFY_(STATUS)
RETURN_(ESMF_Success|ESMF_Failure)
ASSERT_(logical expr)
```

- Profiling

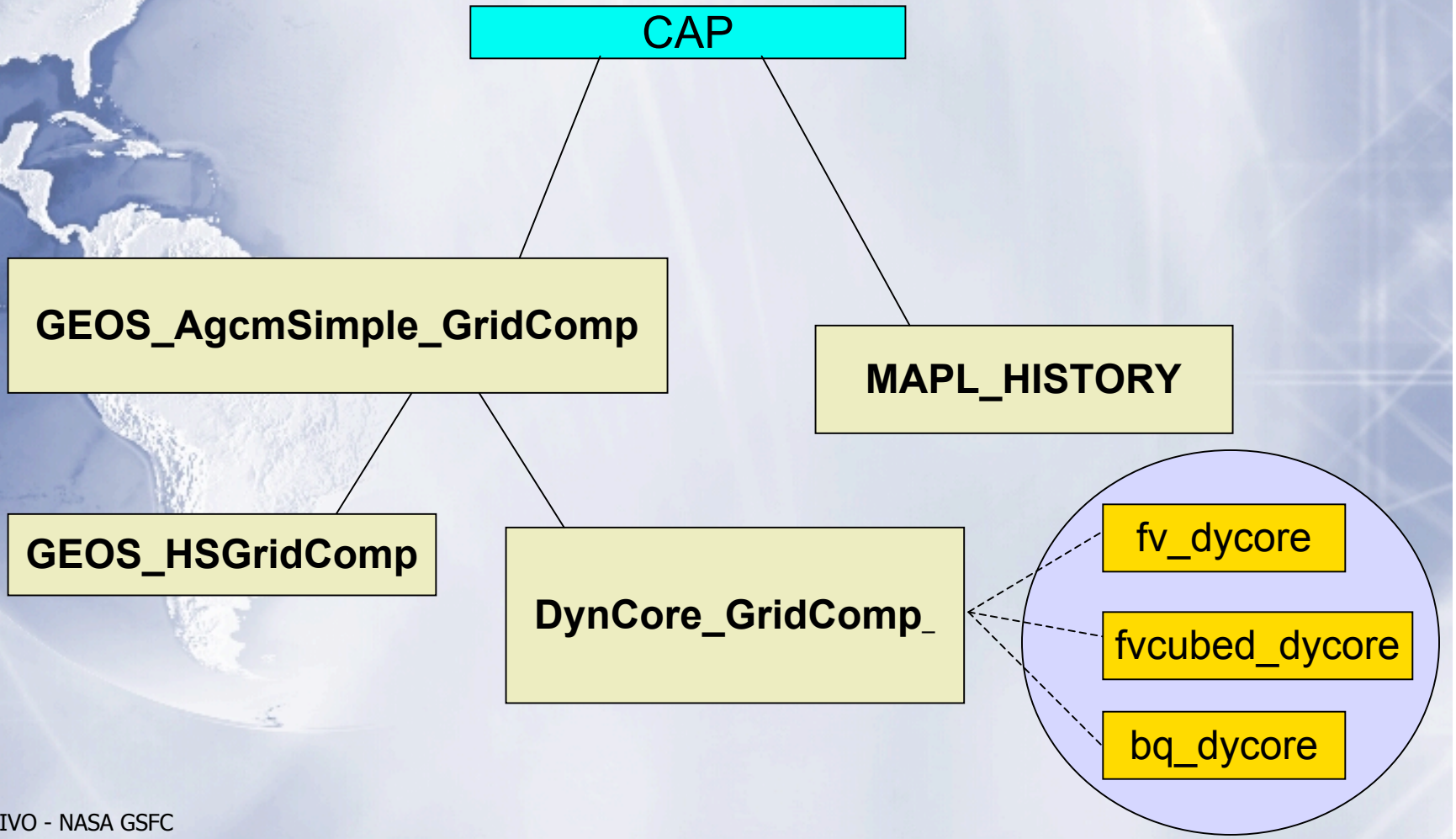
```
MAPL_TimerAdd(MAPL, NAME, RC)
MAPL_TimerOn (MAPL, NAME, RC)
MAPL_TimerOff(MAPL, NAME, RC)
```

- Astronomy
  - MAPL_SunOrbit, MAPL_SunGetInsolation
  - Configuration defines orbital parameters
    - Eccentricity, Perihelion, Obliquity, Equinox
- Universal Constants
  - PI, Grav, Radius, Omega, Rgas, Kappa …

Basic Recipe for a MAPL Component

- Write a SetServices that:
  - Describes the three State's contents
  - Registers any custom IRF methods with MAPL
  - Registers its children with MAPL
  - Explicitly describes connections between children
  - Calls Generic_SetServices
- Write your custom IRF methods
  - calling the generic versions from within each

The DynCore_GridComp Example



The DynCore_GridComp Example

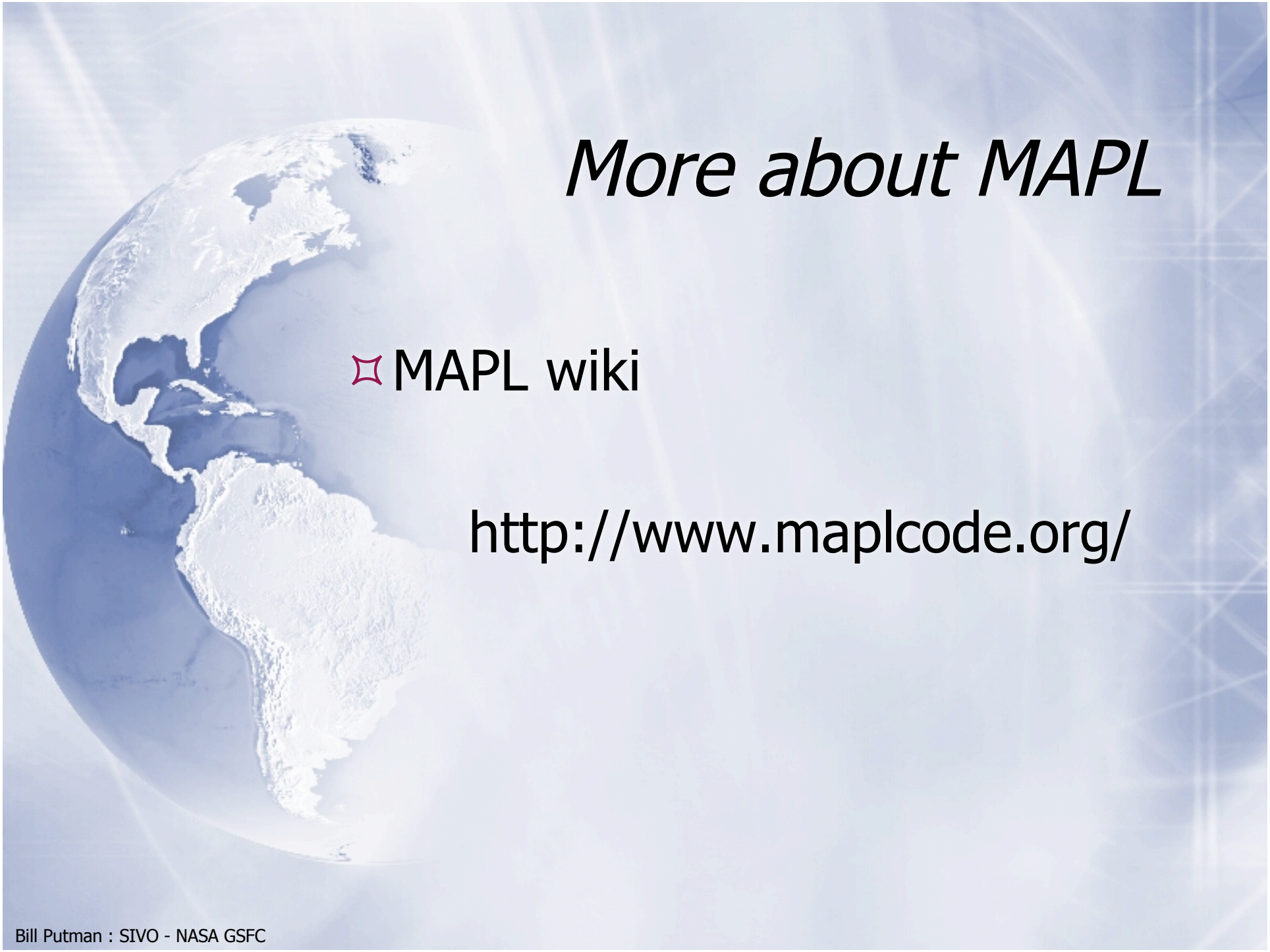
- Receives the ESMF Grid defined in the CAP
  - Cubed-sphere (IMx6*JM) or lat-lon (IMxJM)
- Import State
  - DUDT, DVDT, DTDT, DPEDT, PHIS, QTR
- Internal State
  - U, V, PT, PE, PKZ
- Export State
  - Tendencies, Vorticity, Omega, SLP, Heights, Mass Fluxes and many more…
- Registered Methods
  - Init, Run, RunAddIncs, Finalize, Coldstart

The DynCore_GridComp Example

- Read Restarts during SetServices
  - From File if provided
  - Coldstart routine (if restarts not available)
    - For colloquium we use Coldstart to interface with testcase initializations
    - Configuration files are processed to determine testcase
- Provides basic Initialize Run and Finalize methods
  - These interface with an F90 module defined to tie the ESMF/MAPL layer to the user code
  - Can run adiabatically (dycore only), or connected to simple forcings (Held Suarez), or full GEOS physics components

The DynCore_GridComp Example

```
#if defined(USE_FVcubed)
! FV Specific Module
  use FV_StateMod, only : DynTracers     => T_TRACERS,       &
                          DynVars        => T_FVDYCORE_VARS,  &
                          DynGrid        => T_FVDYCORE_GRID,  &
                          DynState       => T_FVDYCORE_STATE, &
                          DynInit        => FV_InitState,     &
                          DynRun         => FV_Run,           &
                          DynFinalize    => FV_Finalize,      &
                          getAgridWinds  => fv_getAgridWinds, &
                          getOmega       => fv_getOmega,      &
                          getPK          => fv_getPK,         &
                          getVorticity   => fv_getVorticity,  &
                          Agrid_To_Native => a2d3d
#endif
```



More about MAPL

- MAPL wiki

http://www.maplcode.org/