

The Model Coupling Toolkit



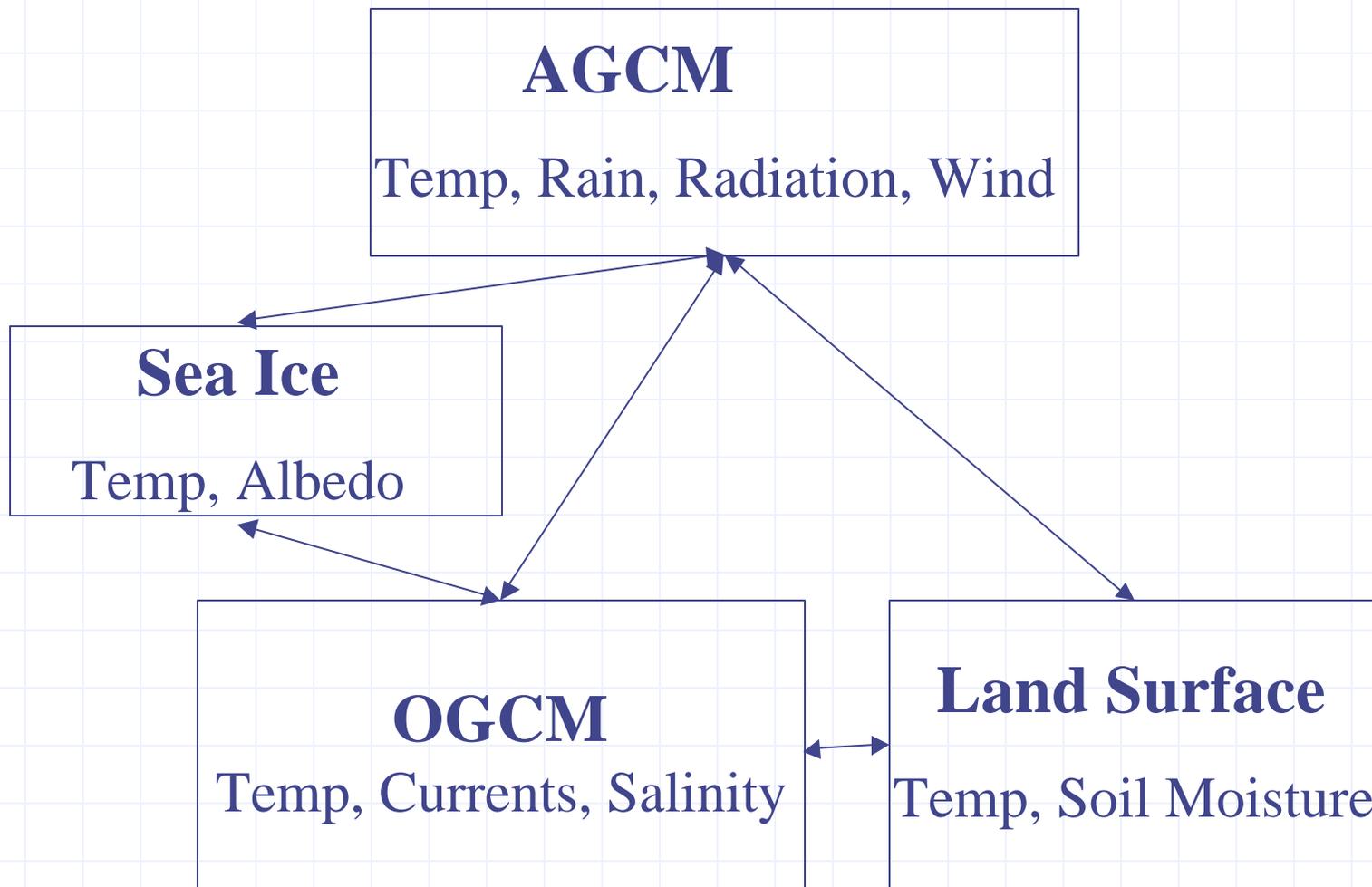
*

Jay Larson and Rob Jacob

Outline

1. Introduction—Coupled Earth System Models
2. Overall Design of the new Community Climate System Model (CCSM) Next-Generation Coupler (NGC)
3. Description of the Model Coupling Toolkit

Typical Coupled Model



General Obstacles to Coupling

◆ Resolution of (usually fixed) numerical grid

- Atmosphere and Land Surface: 2-4 degrees
- Ocean/Sea Ice: 1-2 degrees
- Each model might be on a different grid

◆ Timesteps

- Atmosphere, Land Surface, Sea Ice: .5--1 hour
- Ocean: 6 hours -- 1 day
- Each model might have a different timestep

◆ Code History

- Each model is a Fortran “dusty deck” with different amounts of dust.

Coupling Parallel Models

◆ How do you assign processors to models?

- Distinct processors for each model
 - ◆ load balancing? Exactly how many for each model?
 - ◆ Models which share a grid may be on different numbers of processors.
- All models on the same physical processors
 - ◆ May increase total execution time since each model must execute in turn.
 - ◆ Optimum number of processors for one model may push another model into regime where intra-model communication dominates computation.

◆ Given above choice, how to transfer data between models.

- Trivial Solution: Always gather to one node first.

Current Coupler Architectures

- ◆ Parallel Climate Model—message passing parallel model, with component models and coupler executed as an event loop
- ◆ CCSM—Multiple load image mixed-mode parallel program; components run asynchronously and the coupler is only shared-memory parallel

DOE's Motivation

- ◆ US Climate Researchers have faced problems exploiting microprocessor-based parallel systems to achieve high performance for their applications
- ◆ This situation impedes science efforts:
 - National assessment climate simulations
 - Ongoing climate system model development and testing

ACPI Avant Garde

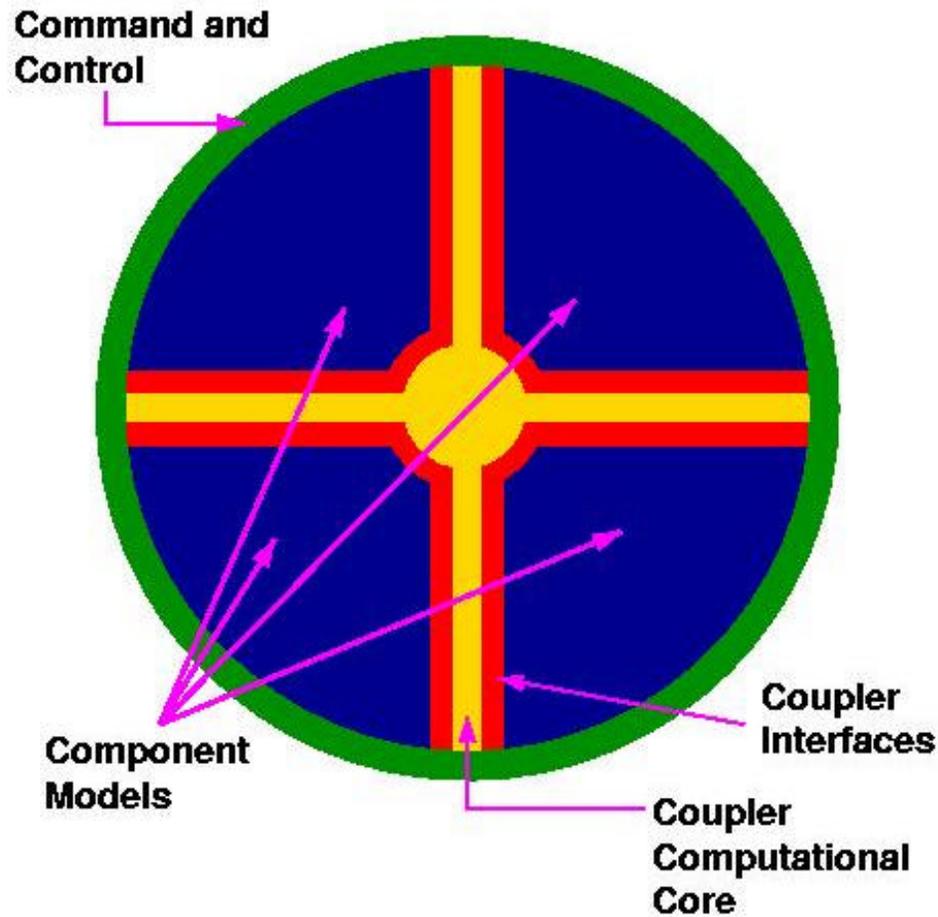


- ◆ Begun June, 2000, this project is a joint enterprise between DOE laboratories and NCAR to develop the next-generation Community Climate System Model (CCSM):
 - High-performance, multiple dynamical core, object-oriented (OO) atmosphere
 - High-performance, OO flux coupler
 - Optimization / Enhancement of other model components
 - Parallel I/O

Our Objectives

- ◆ Create a model coupling environment that is:
 - Flexible
 - Extensible
 - Performance Portable—supports message-passing, shared-memory, and hybrid parallelism
 - Highly configurable and easy-to-use

Coupled Modeling System



The Flux Coupler

- ◆ The Coupler has two basic functions:
 - Command/Control
 - Data flow between component models

We need to support both functions, but need solutions that are highly extensible and configurable

Assumptions

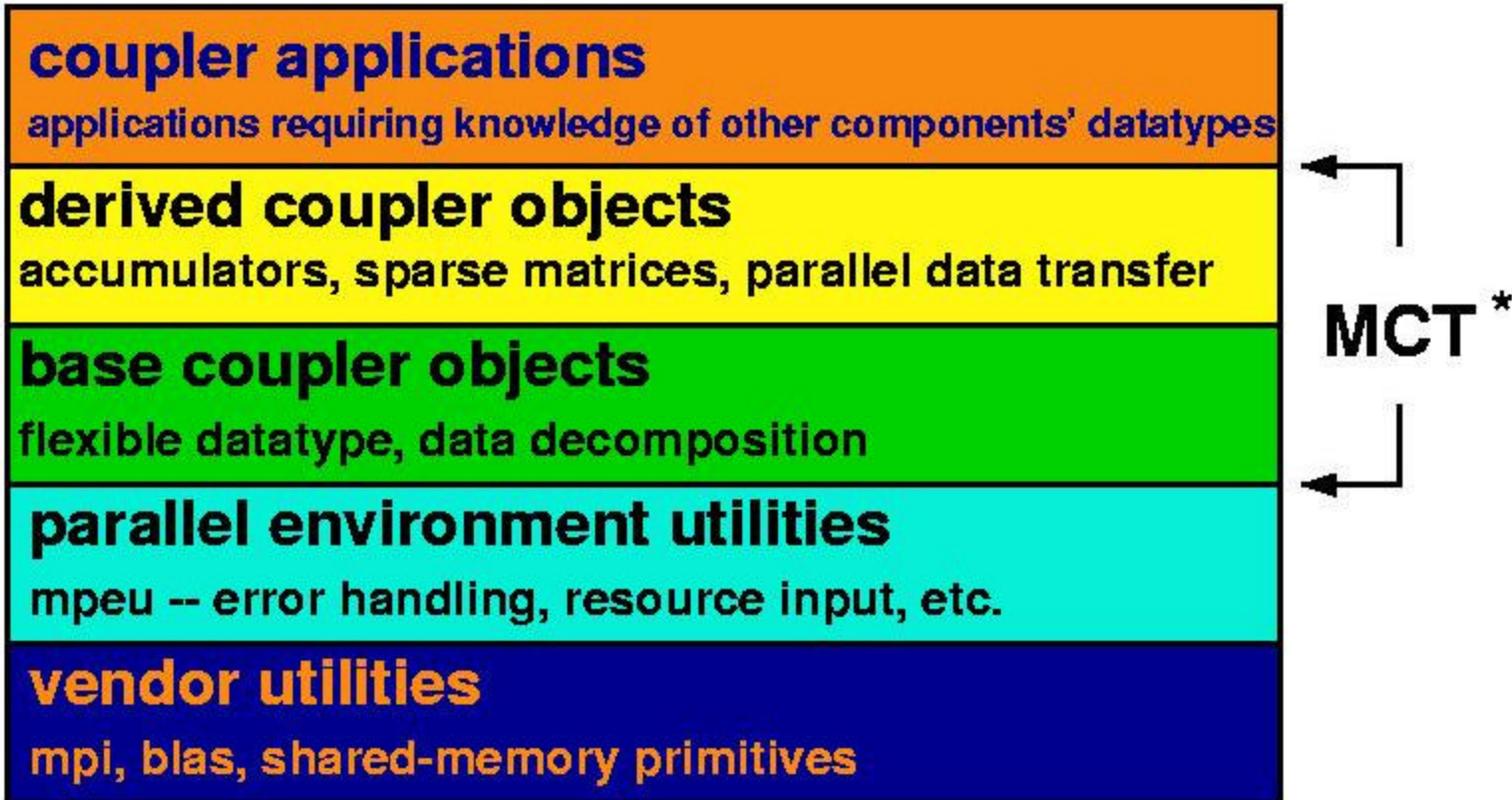
- ◆ Only real and integer data is passed between component models
- ◆ Field data passed to the coupler is represented as vectors
- ◆ Regridding is implemented as sparse matrix-vector multiplication with matrix elements computed off-line

Layered Design:

Listed from Lowest to Highest Level:

- ◆ Vendor utilities (MPI, BLAS, shared-memory)
- ◆ Message Passing Environment Utilities (mpeu), Spherical Coordinate Remapping and Interpolation Package (SCRIP)
- ◆ Model Coupling Toolkit (mct), Message-Passing Handshaking (MPH)
- ◆ Coupler Applications

Next Generation Coupler Design Layers



* Model Coupling Toolkit

mpeu

Provides the following services:

- ◆ F90 module-style access to MPI
- ◆ Portable/Flexible definition of types
- ◆ Support for multiprocessor stdout/stderr
- ◆ Error handling / shutdown
- ◆ Timing/Load balance monitoring tools
- ◆ Sorting Tools
- ◆ Support for basic derived types on which low-level classes in the mct are built

MPH

Multiple Program-Components Handshake Utility

- ◆ F90 module which provides support for carving up MPI_COMM_WORLD among different models.
- ◆ Supports different processor allocation strategies.
- ◆ Each model gets its own communicator.
- ◆ Functions allow each model to find which processors the others are running on.
- ◆ Written by Chris Ding and Helen Ye of LBNL



Basic, low-level classes:

Data decomposition—**GlobalMap**,
GlobalSegMap

Internal data representation—**AttrVect**

Indexing—**Navigator**



Coupler functionality classes:

- ◆ Grid representation—**GeneralGrid**
- ◆ Transformation Matrices—**SparseMatrix**
- ◆ Time Sum/Average—**Accumulator**
- ◆ Flux Merging—**MergeMask**
- ◆ Parallel Data Transfer and Transpose
- ◆ Diagnostics / Global and Hemispheric averages



Coupler application classes:

Couplings between component models—
Contract

The **Contract** encapsulates all the lower-level data types used to effect transfer and conversion of data between components.

Once supported, it will be possible to create a coupled modeling system with component models and connections all specified at run-time



1-D Contiguous Data Decomposition—the **GlobalMap**

Type GlobalMap

integer :: comm

integer :: gsize

integer :: counts(:)

integer :: displs(:)

End Type GlobalMap

Methods—create, destroy, *et cetera*



1-D Segmented Data Decomposition—the **GlobalSegMap**

Type GlobalSegMap

```
integer                :: comm  
integer                :: ngseg  
integer                :: gsize  
integer                :: lsize  
integer, dimension(:) :: start  
integer, dimension(:) :: length  
integer, dimension(:) :: pe_loc
```

End Type GlobalSegMap

Suitable for describing a 2-D decomposition of a
grid



1-D Internal Data Representation— the *Attribute Vector* **AttrVect**

```
Type AttrVect
```

```
  type(List) :: iList
```

```
  type(List) :: rList
```

```
  integer, dimension(:, :), pointer ::  
  iAttr
```

```
  real, dimension(:, :), pointer :: rAttr
```

```
End Type AttrVect
```



AttrVect Methods—

- ◆ Create, Destroy
- ◆ Count and Reference Attributes
- ◆ Scatter, Gather, Broadcast
(implemented in a separate module)
- ◆ Sort, Permute, SortPermute



AttrVect Example—store n values of
2m winds and temperature in
AttrVect named `av_2m`:

Tags—tag 2m zonal wind as `u2m`, 2m
meridional wind as `v2m`, and 2m
temperature as `t2m`.

Call `AttrVect_init(av_2m,rList='u2m:v2m:t2m',n)`



Initialization:

```
Call AttrVect_init(av_2m,rList='u2m:v2m:t2m',n)
```

Indexing/Access:

```
Index_t2m = AttrVect_indexRA(av_2m,'t2m')
```

2m temperature stored in
`av_2m%rAttr(index_t2m,:)`



Local Indexing—the **Navigator**
(built on top of the **AttrVect**
class)



Unstructured Multidimensional Grid Representation—the **GeneralGrid**

Type GeneralGrid

`type(List) :: coordinates`

`type(List) :: weights`

`type(List) :: ordering`

`type(AttrVect) :: points`

End Type GeneralGrid

Methods—create, destroy, sort—all built upon
AttrVect methods



Time Sum/Average—the Accumulator

Type Accumulator

```
integer :: num_steps
```

```
integer :: steps_done
```

```
type(AttrVect) :: av
```

End Type Accumulator

Methods—create, destroy, sort...and of course accumulation!



Linear Transformations—the SparseMatrix

◆ Built on AttrVect class, with restricted attributes. For SparseMatrix variable **S_mat**

- `Smat%iList = 'row:column'`
- `Smat%rList = 'weight'`

◆ Inherits all AttrVect methods

Proposed Parallel Transfer Algorithm

Component1

Component2

MPH provides nprocs1 and nprocs2 for allocating memory
Describe decomposition with GlobalMap
Each Component is on the same grid.
Grid points are numbered
Each processor handles a range of points.
Exact range determined at runtime given nprocs1 and nprocs2

Gather local ranges to root
Send to Component1

Receive Component2 ranges on root
Broadcast to other Component1 processors

Use local range and Component2 local ranges
to determine which Component2 processors must
have data sent to them.

Store this information in SendMap
SendMap: for each processor on Component2,
provide list of points to send

Gather SendMaps to root and send
to Component2

Receive SendMaps
on root and broadcast

Send data to Component2

Use SendMap to determine
which Component1
processors to receive from



Intermodel couplings—the Contract

Type Contract

```
logical          :: new
type(String)    :: partner
type(String)    :: operation
integer         :: frequency
type(GlobalSegMap) :: remote_map
type(GlobalSegMap) :: local_map
type(GeneralGrid) :: local_grid
type(GeneralGrid) :: remote_grid
..
type(AttrVect)  :: data
```

End Type Contract

Goal—each component model is initialized with an array of Contracts, which govern inter-component data transfer and conversion



Status of the mct:

- ◆ Numerous modules containing classes and their methods have been implemented
- ◆ Lower-level class/method APIs released
- ◆ Coupler functionality class/method APIs under construction
- ◆ Initial (alpha) Toolkit release expected February 2001

Potential Future Applications

- ◆ Coupling of a 0-D box atmospheric chemistry model to CCSM
- ◆ Direct coupling of CCSM to MM5 to provide better temporal resolution of boundary conditions for MM5
- ◆ Regional Coupled Modeling System
- ◆ Starting point for portions of Earth System Modeling Framework (ESMF)