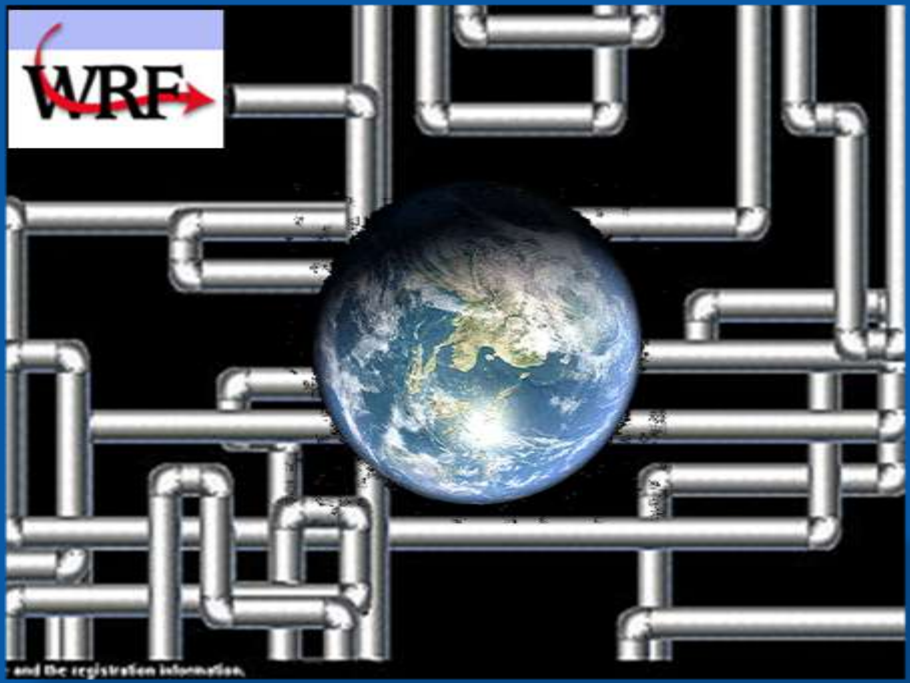
NREL
NATIONAL RENEWABLE ENERGY LABORATORY

Coupling WRF to Other Models



John Michalakes

WRF Users Workshop

June 25, 2010

Purpose

Provide users with introduction to…

- Why couple WRF to other models?
- Modes of coupling and tradeoffs
- WRF infrastructure provisions

Scope

High level

- Coupling is complex, app. specific
- Multi-faceted – complexity is a product of:
  - Each component's complexity
  - Infrastructure's complexity
  - Complexity of system as a whole

1 hour tutorial…

- Only covering WRF side with some use cases
- Where to find out more (at the end)

Model Coupling Overview

Why couple?

Types of coupling

Coupling packages

Coupling support in WRF software

Examples

- WRF-HyCOM using MCEL
- WRF-HyCOM using ESMF

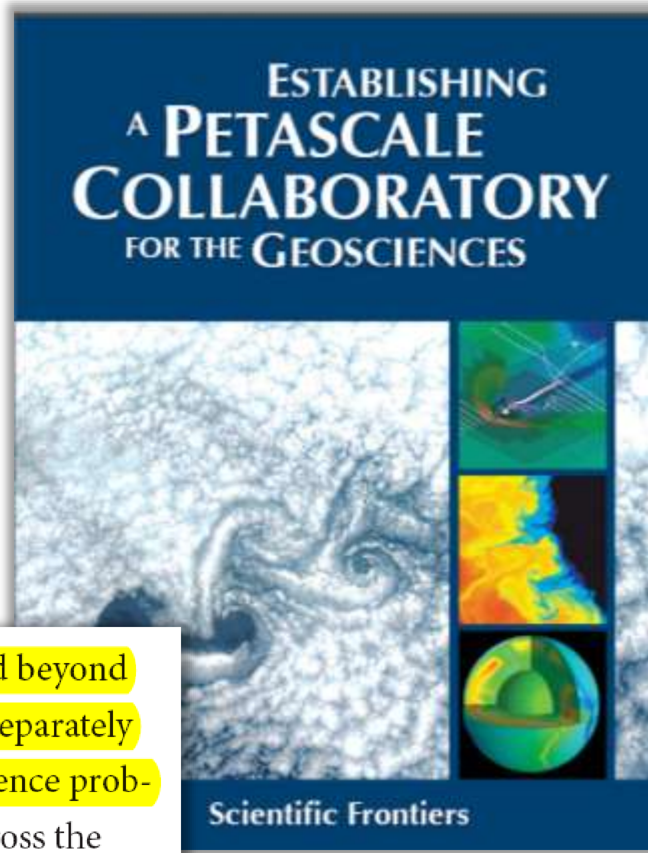
Final thoughts

Why couple?

We are past the point of single model simulations.

Trend will be multi-scale multi-model simulations.

In the past decade, the geosciences have progressed beyond traditional disciplinary organization that focused separately on atmospheric, oceanic, solid Earth and space science problems. The challenges today demand integration across the disciplinary science because the individual systems and the

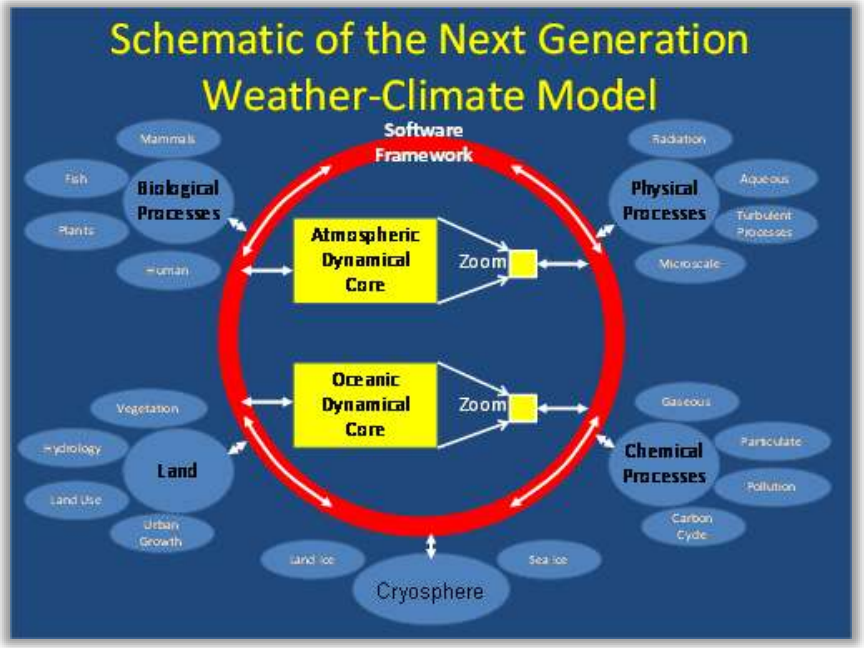


Ad Hoc Committee and Technical Working Group for a Petascale Collaboratory for the Geosciences. 2005. *Establishing a Petascale Collaboratory for the Geosciences: Scientific Frontiers.* A Report to the Geosciences Community. UCAR/JOSS. 80 pp.

Why couple?

We are past the point of single model simulations.

Trend will be multi-scale multi-model simulations.



Wind Energy (MCEL)

Challenges

- Complex physical phenomena
  - Anisotropic turbulent flow
  - Wave, wind and turbine interaction
  - Dynamic system, many degrees of freedom
  - Large range of scales
- Complex energy resource
  - Large variation in loads
  - Highly unpredictable

WP1 – wind and ocean condition

Leader: Idar Barstad at the Bjerknes Centre, (idar.barstad@bjerknes.uib.no)

Coupled numerical model system (scale: 100km-10m):

- Mesoscale model – larger & medium scales
- Wave model for ocean waves
- Computational Fluid Dynamics (CFD) model for microscale

An integrated system describing geophysical conditions vital for offshore wind energy.

HyWind Floating Turbine

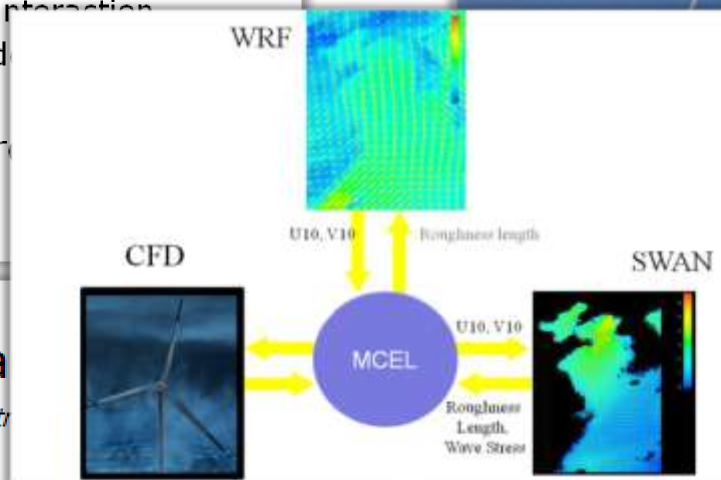
norcowe
Norwegian Centre for Offshore Wind Energy

Slides and images courtesy Idar Barstad

Wind Energy (MCEL)

Challenges

- Complex physical phenomena
  - Anisotropic turbulent flow
  - Wave, wind and turbine in
  - Dynamic system, many d
  - Large range of scales
- Complex energy resour
  - Large variation in loads
  - Highly unpredictable



WP1 – wind and ocea

Leader: Idar Barstad at the Bjerknes Centr

Coupled numerical model system (scale: 100km-10m):

- Mesoscale model – larger & medium scales
- Wave model for ocean waves
- Computational Fluid Dynamics
  (CFD) model for microscale

An integrated system describing geophysical conditions vital for offshore wind energy.

HyWind Floating Turbine

norcowe
Norwegian Centre for Offshore Wind Energy

Slides and images courtesy Idar Barstad

Model Coupling: Coupling Modes

Subroutinized

- Communication through subroutine calls and argument lists
- Components must be code-compatible
- More efficient

Model Coupling: Coupling Modes

Subroutinized

- Communication through subroutine calls and argument lists
- Components must be code-compatible
- More efficient

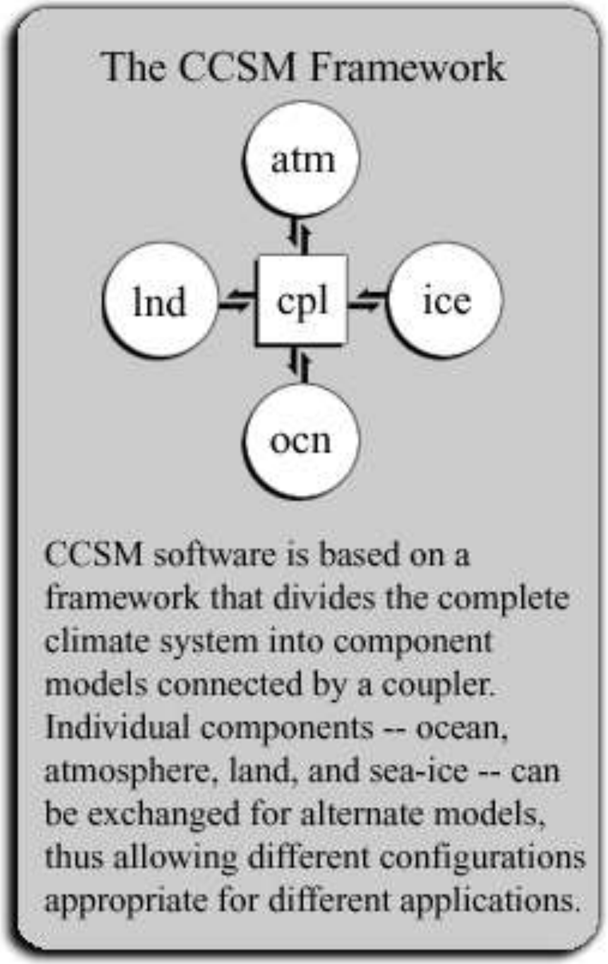
Componentized

- Component interaction: Scheduled or Peer-to-Peer
- Component execution: Sequential or Concurrent
- More flexible

Component Interaction

Scheduled

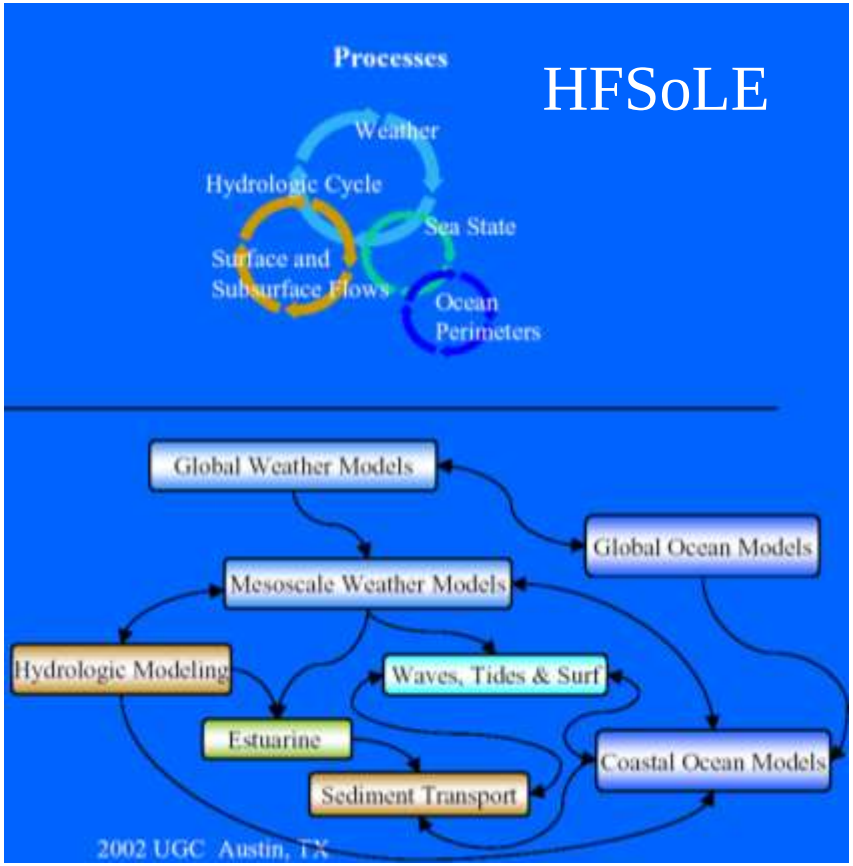
- Regular, a-priori scheduled interactions
- Logically "hub and spoke"
- May have distinct coupler component (but not necessary)
- Example: Community Climate System Model (CCSM)
  - CAM, POP, CICE, LSM
  - Coupler implemented using MCT



Component Interaction

Peer-to-peer

- Logically federated
- Components produce and consume
- Components may comprise schedule-coupled components
- No central control or schedule, essentially data-flow
- Example: HFSoLE
  - Atmosphere: COAMPS or WRF
  - LSOM, SWAN, ADCIRC (Persian Gulf)
  - NCOM, SWAN, LSOM (Adriatic)
  - ADH, NCOM, ADCIRC (Mississippi Sound)
  - Coupling infrastructure using MCEL

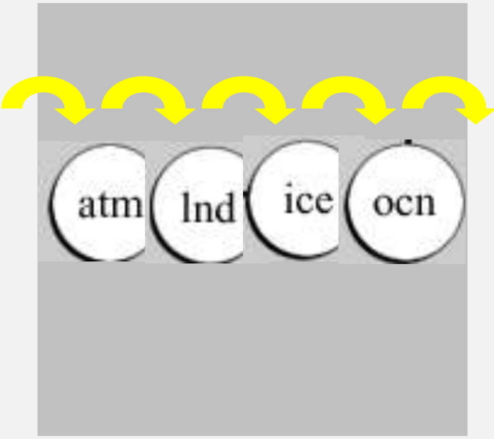


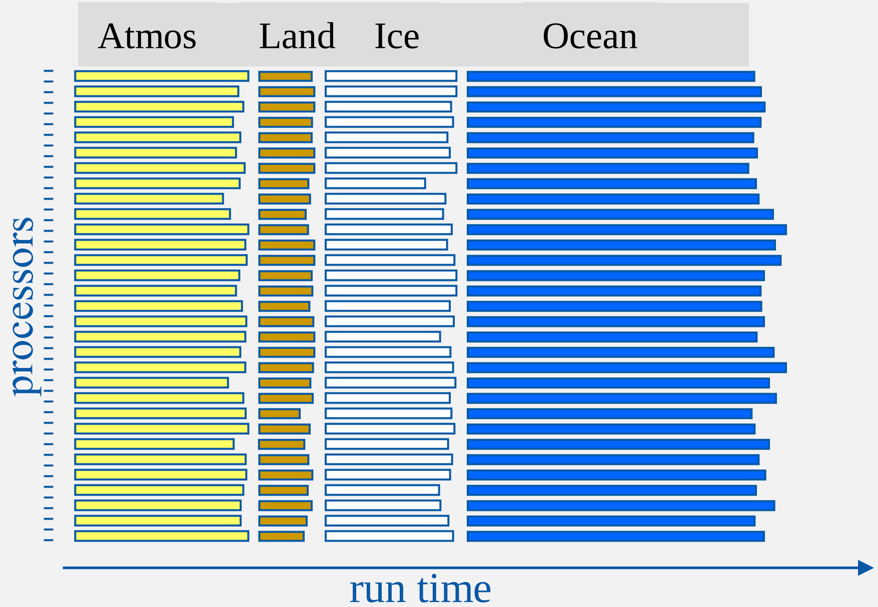
Source: Rick Allard, NRL, UGC 2002 Presentation Slides

Types of Coupling

Sequential coupling

– Components are run in sequence by all processors
– Forcing data is interpolated from grid-to-grid
– All processes can be kept active without forcing components to be out of phase with each other
– Performance and scaling is limited by least scalable component

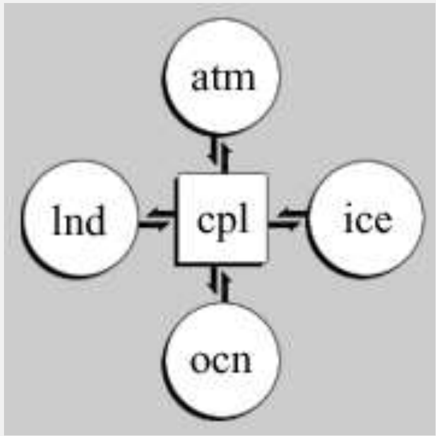


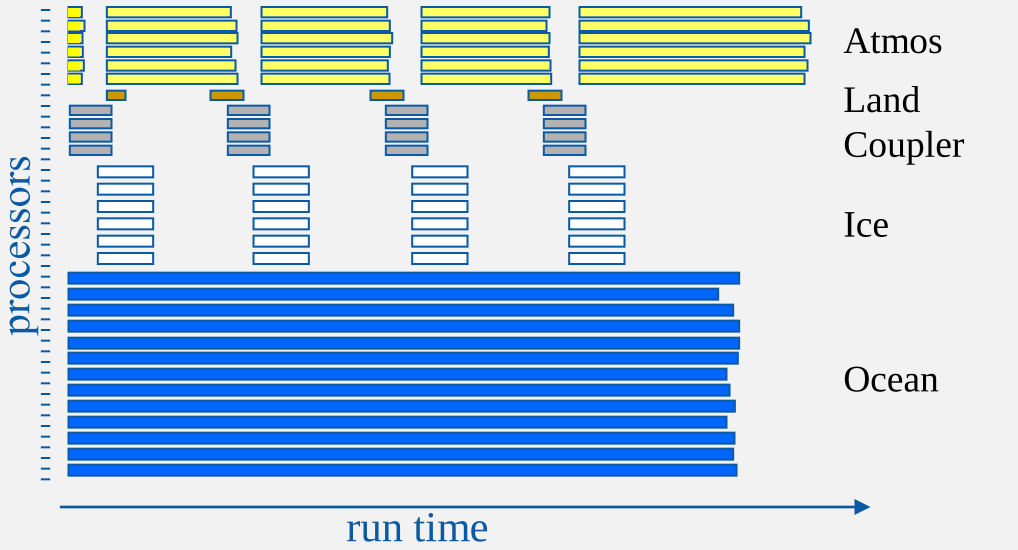


Types of Coupling

Concurrent coupling

- Components integrate concurrently on separate sets of processes
- Periodically communicate forcing data to other components on some schedule
- Parallelism is both within and between the components; subject to load imbalance
- Two-way coupling requires solutions from components to be slightly out of phase if the components are to run concurrently





Types of Coupling

Hybrid concurrent and sequential

- Components with sequential dependency or small components execute sequentially filling in the gaps
- Large components run concurrently
- Latest version of CCSM will run this way

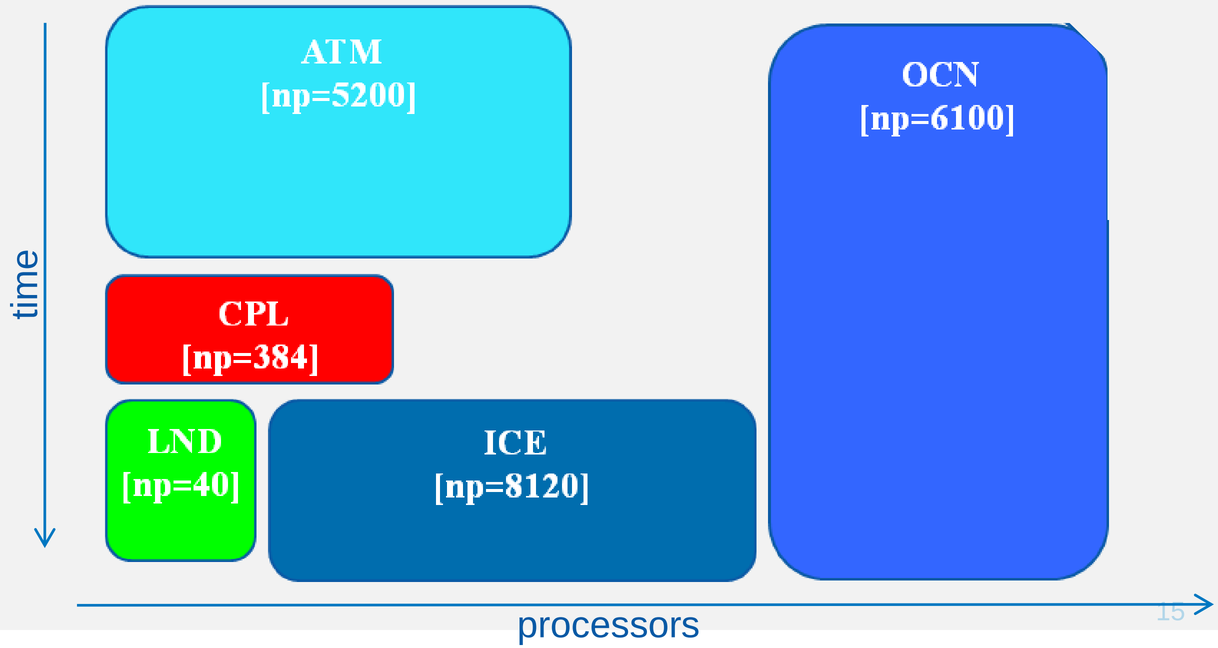


Figure courtesy of Rich Loft, NCAR

Coupling packages

Attributes

- Functionality
- Efficiency
- Portability
- Flexibility
- Ease of deployment/use/maintenance
- Support and adoption
- Status in WRF

Available packages

- ESMF (NOAA)
  - Large community effort
  - Basis for NOAA modeling systems
  - WRF one of the first ESMF components
  - www.earthsystemmodeling.org
- MCEL (AFRL)
  - DoD-developed coupler
  - U. Miami, NORCOWE, other WRF users
  - Not scalable
  - http://www.bettencourt.info/MCEL
- MCT (Argonne NL)
  - Basis for CCSM coupler
  - WRF implementation is tot current supported to community but being revived with regional climate modeling at NCAR
  - http://www.mcs.anl.gov/research/projects/mct/
- HRF Coupler
  - Specific to HWRF and configuration dependent

	Functionality	Efficiency	Portability	Flexibility	Ease of Use	Support and Adoption	In WRF
Subroutine	Sequential Single-exe No regridding	Very good Virtually no overhead for coupling Scales as well as app itself	As portable as app itself	Plug and play support from WRF Framework Model Layer Interface	As easy or difficult as adding a subroutine to WRF.	One-off implemenation. No support; no adoption.	Supported by WRF Software Model Layer Interface
MCEL	Concurrent Multi-exe Data driven On-line & off-line Run-time regridding Conservative possible	Adequate for 2D coupling but client-server architecture a scaling bottleneck. Data-driven control may help with automatic load balance.	Needs TCP/IP sockets. Many package dependencies.	Very good. Interaction with other apps looks like I/O.	Difficult to install initially. Transparent and easy to use thereafter.	One person. Some adoption.	Through WRF I/O API
MCT	Sequential/Concurrent Single exe (Multi exe possible) Concurrent/Multi-exec. Offline regridding Conservative	Good performance and scaling	Widely ported.	Flexibility is up to the implementer; MCT does not impose or enforce.	It is a toolkit and must be explicitly programmed. Need to jam apps into single executable adds complexity.	Supported as part of DOE contribution to CCSM. Adopted by other groups.	Through WRF I/O API. Has fallen out of use.
ESMF	Sequential/Concurrent Single exe (Multi exe possible) On-line regridding (non-conserv.) Offline regridding (conserv.) Nesting is problematic.	Good performance and scaling	Widely ported.	Plug and play is suported between components that have been reengineered to ESMF APIs	Coupling must be explicitly programmed. Considerable reengineering required for existing apps. Need to jam different apps into single executables adds complexity.	Very well supported and maintained. Stable funding. Widespread adoption, but at varying and sometimes trivial levels of "compliance"	Coupling to single domain through WRF I/O API

WRF Support for Coupling

WRF does coupling the same way it does I/O

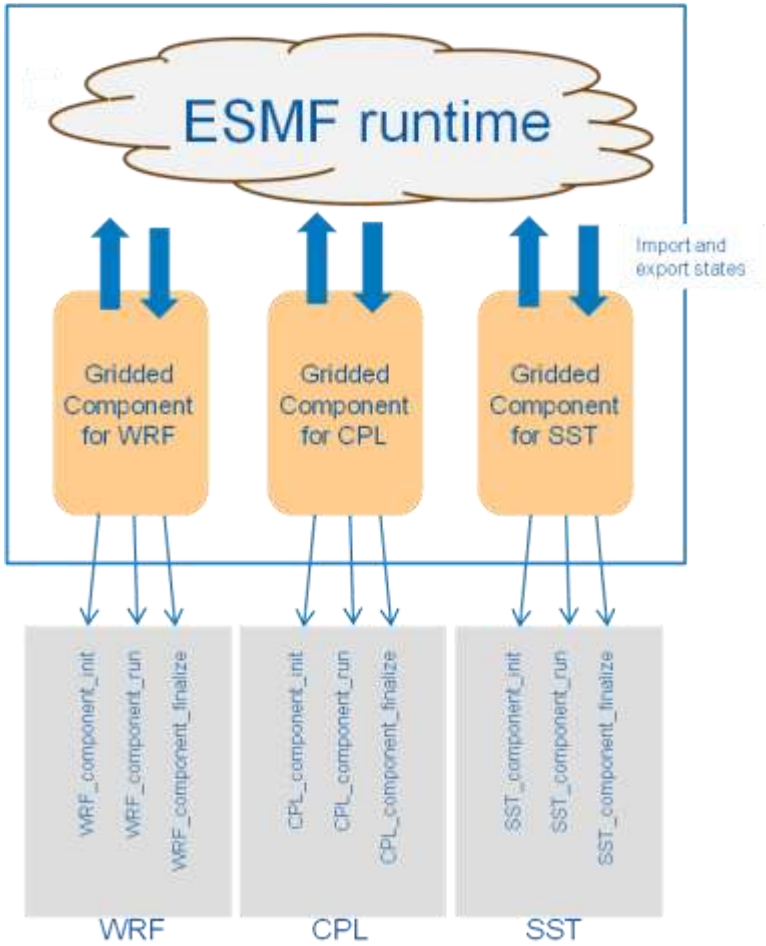
- Output to other models is specified by adding state data to I/O streams in the Registry
- The I/O streams are directed to coupling formats using the io_form setting for the stream in the namelist.input file
- Refer to WRF software tutorial notes

Other components may need to be reprogrammed explicitly to add interfaces to the coupling infrastructure you wish to use (eg. ESMF, MCEL)

Case: WRF and ESMF

WRF-SST Coupled code

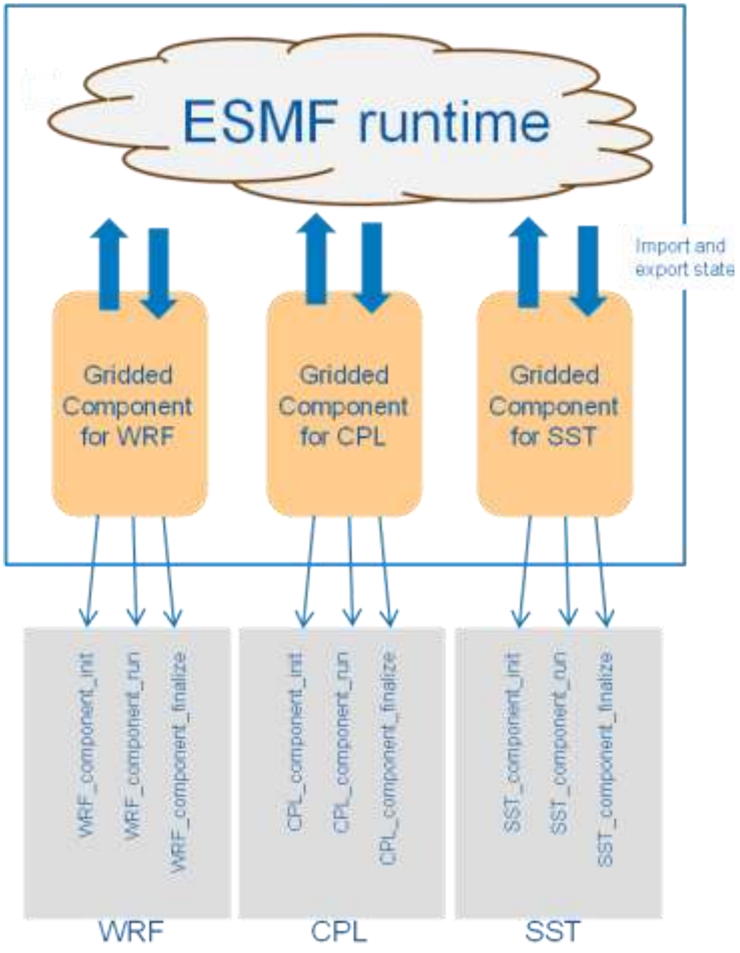
- By Tom Henderson (NOAA) who developed for the ESMF implementation of the WRF I/O API
- Self-contained and distributed with WRFV3
- ESMF Components
  - WRF Model
  - “Data Ocean” Component
  - Coupler
- Simplified
  - All components on same grid (no interpolation)
  - Coupling to a single WRF domain
  - Includes coupling-through-files mode for verification
- This is a good template to start with when coupling WRF to other models through ESMF. The WRF-LIS coupled system was built this way.



Case: WRF and ESMF

WRF-SST Coupled code

- Files and directories:
  - `./main/wrf_SST_ESMF.F`
    - Contains definition of data-ocean and coupler
    - Main program
  - `./main/wrf_ESMFMod.F`
    - Contains definition of ESMF parts of WRF component
  - `./external/io_esmf/README.io_esmf`
    - Detailed instructions for building the test case
  - `./test/em_esmf_exp/README_WRF_CPL_SST.txt`
    - Detailed instructions for running the test case



Case: WRF and ESMF

./main/wrf_SST_ESMF.F

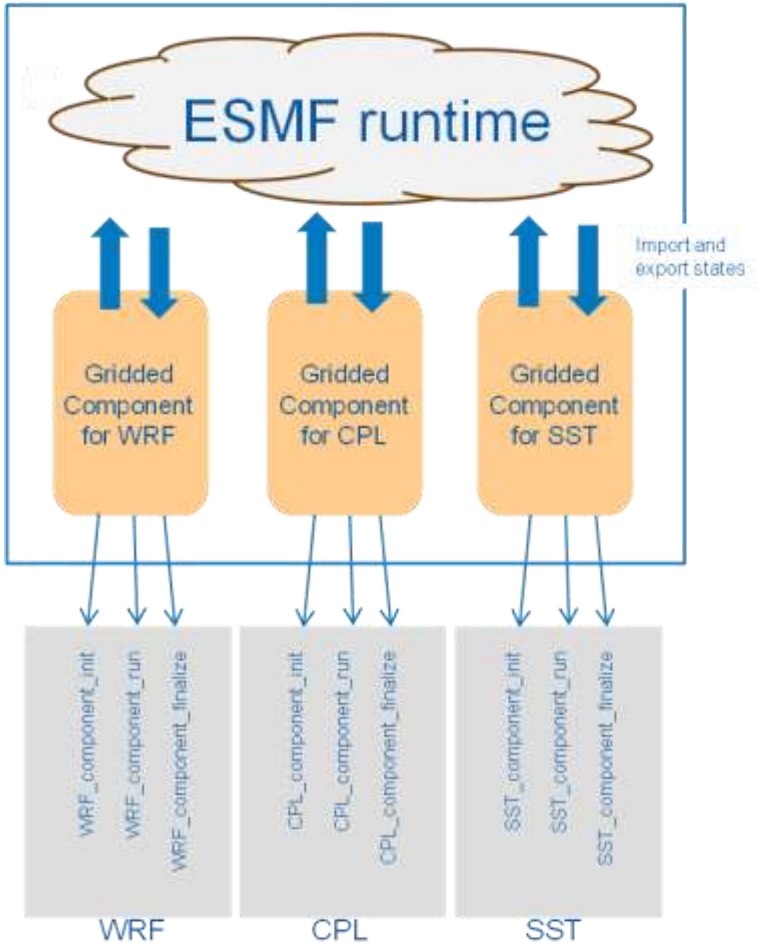
- PROGRAM wrf_SST_ESMF
  - Initializes ESMF
  - Creates instances of the Gridded Components
    - WRF Model
    - SST Dummy Model (this is the "ocean")
    - WRF-SST Coupler
  - "Registers" (ESMF verb) the components by including entry points provided by components that "init", "run" and "finalize"
    - WRF provides wrf_component_run
    - Dummy ocean provides sst_component_run
    - Coupler provides (guess)
  - Creates the import and export states
  - Add fields to the states (not shown)
  - Initialize components adding states
  - Time loop: run each component in sequence
    - Export of SST is Import of CPL
    - Export of CPL is Import of WRF
    - Export of WRF is Import of CPL
    - Export of CPL is import of SST

(if 2-way)

  - Shut down components

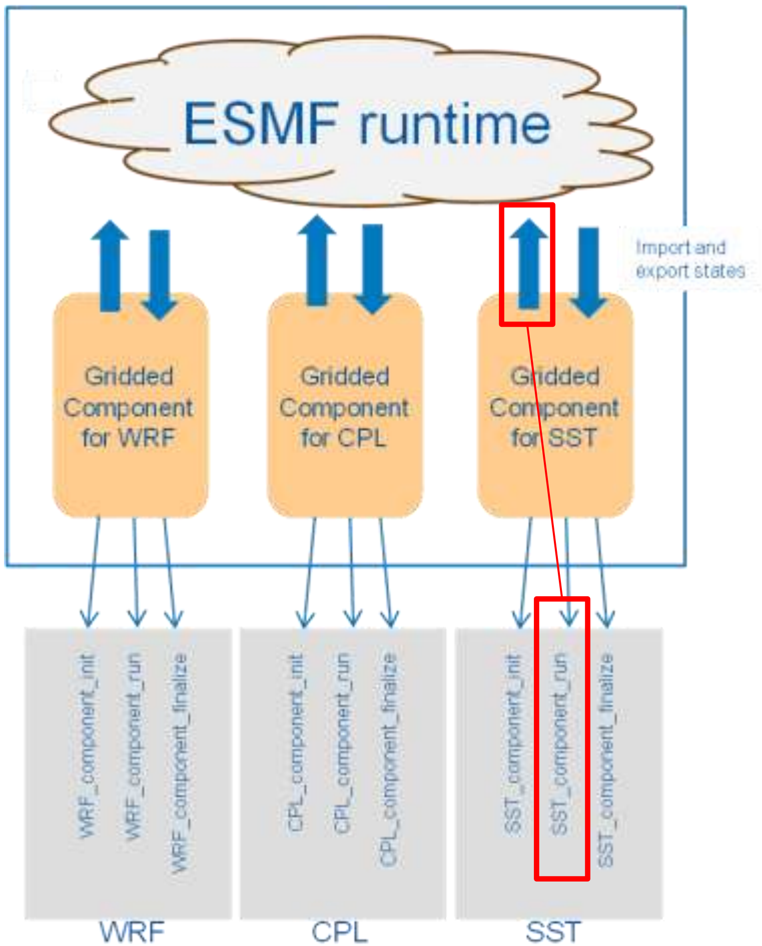
```
PROGRAM wrf_SST_ESMF
 CALL ESMF_Initialize( vm=vm, ...
 compWRF = ESMF_GridCompCreate(name="WRF", …
 compSST = ESMF_GridCompCreate(name="SST", …
 compCPL = ESMF_GridCompCreate(name="CPL", …
 importWRF = ESMF_StateCreate( …
 exportWRF = ESMF_StateCreate( …
 importSST = ESMF_StateCreate( …
 exportSST = ESMF_StateCreate( …
 importCPL = ESMF_StateCreate( …
 exportCPL = ESMF_StateCreate( …
 CALL ESMF_GridCompSetServices(compWRF, WRF_register,
 CALL ESMF_GridCompSetServices(compSST, SST_register,
 CALL ESMF_GridCompSetServices(compCPL, CPL_register,
 CALL ESMF_GridCompInitialize(compWRF,
        importStateWRF, exportStateWRF, clock, …
 CALL ESMF_GridCompInitialize(compSST,
        importStateSST, exportStateSST, clock, …
 CALL ESMF_GridCompInitialize(compCPL,
        importStateCPL, exportStateCPL, clock, …
 DO WHILE ( .NOT. ESMF_ClockIsStopTime(clock))
  CALL ESMF_GridCompRun(compSST,
        importStateSST, exportStateSST, clock, …
  CALL ESMF_CplCompRun(compCPL,
        exportStateSST, importStateWRF, clock, …
  CALL ESMF_GridCompRun(compWRF,
        importStateWRF, exportStateWRF, clock, …
 ENDDO
 CALL ESMF_GridCompFinalize(compSST, …
 CALL ESMF_GridCompFinalize(compWRF, …
 CALL ESMF_GridCompFinalize(compCPL, …
END PROGRAM wrf_SST_ESMF
```

Case: WRF and ES




```
PROGRAM wrf_SST_ESMF
 CALL ESMF_Initialize( vm=vm, ...
 compWRF = ESMF_GridCompCreate(name="WRF", …
 compSST = ESMF_GridCompCreate(name="SST", …
 compCPL = ESMF_GridCompCreate(name="CPL", …
 importWRF = ESMF_StateCreate( …
 exportWRF = ESMF_StateCreate( …
 importSST = ESMF_StateCreate( …
 exportSST = ESMF_StateCreate( …
 importCPL = ESMF_StateCreate( …
 exportCPL = ESMF_StateCreate( …
 CALL ESMF_GridCompSetServices(compWRF, WRF_register,
 CALL ESMF_GridCompSetServices(compSST, SST_register,
 CALL ESMF_GridCompSetServices(compCPL, CPL_register,
 CALL ESMF_GridCompInitialize(compWRF,
        importStateWRF, exportStateWRF, clock, …
 CALL ESMF_GridCompInitialize(compSST,
        importStateSST, exportStateSST, clock, …
 CALL ESMF_GridCompInitialize(compCPL,
        importStateCPL, exportStateCPL, clock, …
 DO WHILE ( .NOT. ESMF_ClockIsStopTime(clock))
  CALL ESMF_GridCompRun(compSST,
        importStateSST, exportStateSST, clock, …
  CALL ESMF_CplCompRun(compCPL,
        exportStateSST, importStateWRF, clock, …
  CALL ESMF_GridCompRun(compWRF,
        importStateWRF, exportStateWRF, clock, …
 ENDDO
 CALL ESMF_GridCompFinalize(compSST, …
 CALL ESMF_GridCompFinalize(compWRF, …
 CALL ESMF_GridCompFinalize(compCPL, …
END PROGRAM wrf_SST_ESMF
```

Case: WRF and ESMF




```
PROGRAM wrf_SST_ESMF
 CALL ESMF_Initialize( vm=vm, ...
 compWRF = ESMF_GridCompCreate(name="WRF", …
 compSST = ESMF_GridCompCreate(name="SST", …
 compCPL = ESMF_GridCompCreate(name="CPL", …
 importWRF = ESMF_StateCreate( …
 exportWRF = ESMF_StateCreate( …
 importSST = ESMF_StateCreate( …
 exportSST = ESMF_StateCreate( …
 importCPL = ESMF_StateCreate( …
 exportCPL = ESMF_StateCreate( …
 CALL ESMF_GridCompSetServices(compWRF, WRF_register,
 CALL ESMF_GridCompSetServices(compSST, SST_register,
 CALL ESMF_GridCompSetServices(compCPL, CPL_register,
 CALL ESMF_GridCompInitialize(compWRF,
        importStateWRF, exportStateWRF, clock, …
 CALL ESMF_GridCompInitialize(compSST,
        importStateSST, exportStateSST, clock, …
 CALL ESMF_GridCompInitialize(compCPL,
        importStateCPL, exportStateCPL, clock, …
 DO WHILE ( .NOT. ESMF_ClockIsStopTime(clock))
  CALL ESMF_GridCompRun(compSST,
        importStateSST, exportStateSST, clock, …
  CALL ESMF_CplCompRun(compCPL,
        exportStateSST, importStateWRF, clock, …
  CALL ESMF_GridCompRun(compWRF,
        importStateWRF, exportStateWRF, clock, …
 ENDDO
 CALL ESMF_GridCompFinalize(compSST, …
 CALL ESMF_GridCompFinalize(compWRF, …
 CALL ESMF_GridCompFinalize(compCPL, …
END PROGRAM wrf_SST_ESMF
```

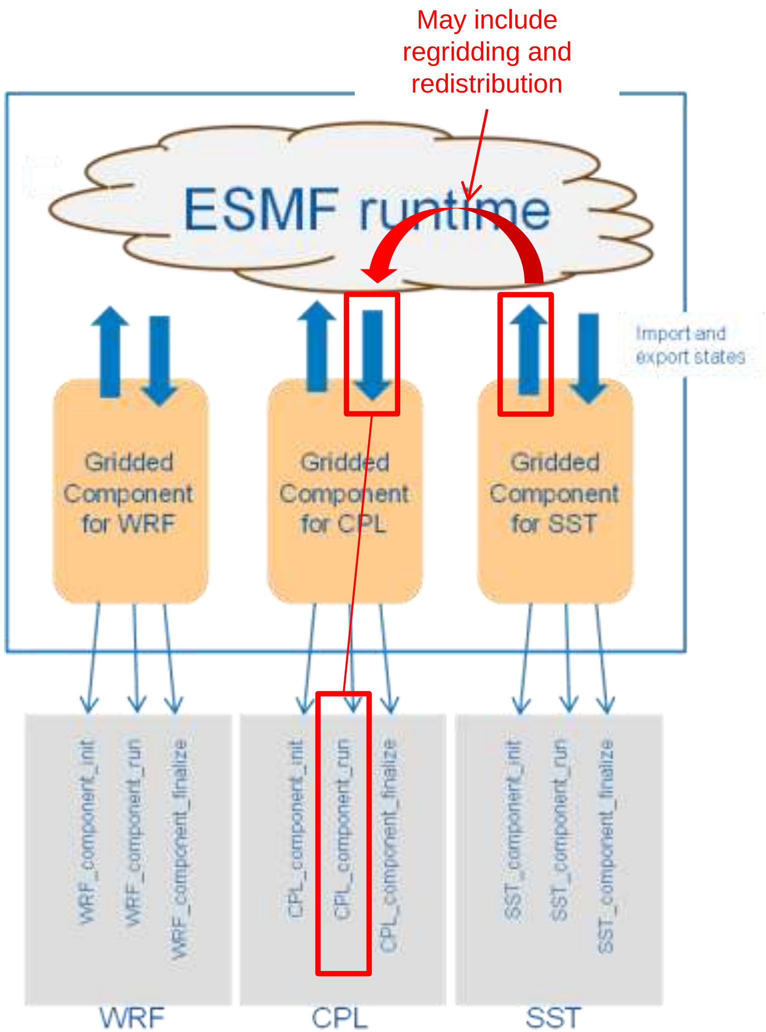
Case: WRF and ESMF

./main/wrf_SST_ESMF.F

- SUBROUTINE sst_component_run
  - Reads data from a file into a temporary array named inptr
  - Put data into ESMF export state
    - Created in earlier example
    - Defined as collection of fields (not shown)
    - FIELD consists of
      - Name
      - Data, here a 2-D array allocated when field was defined (not shown)
      - Other attributes (not used in this example)
  - Access a field from the state and store field in a temporary of type ESMF_Field
  - Access the array from the field by passing in a pointer. On return *ptr* will point to the 2D array stored in the field
  - Copy the data that was read into

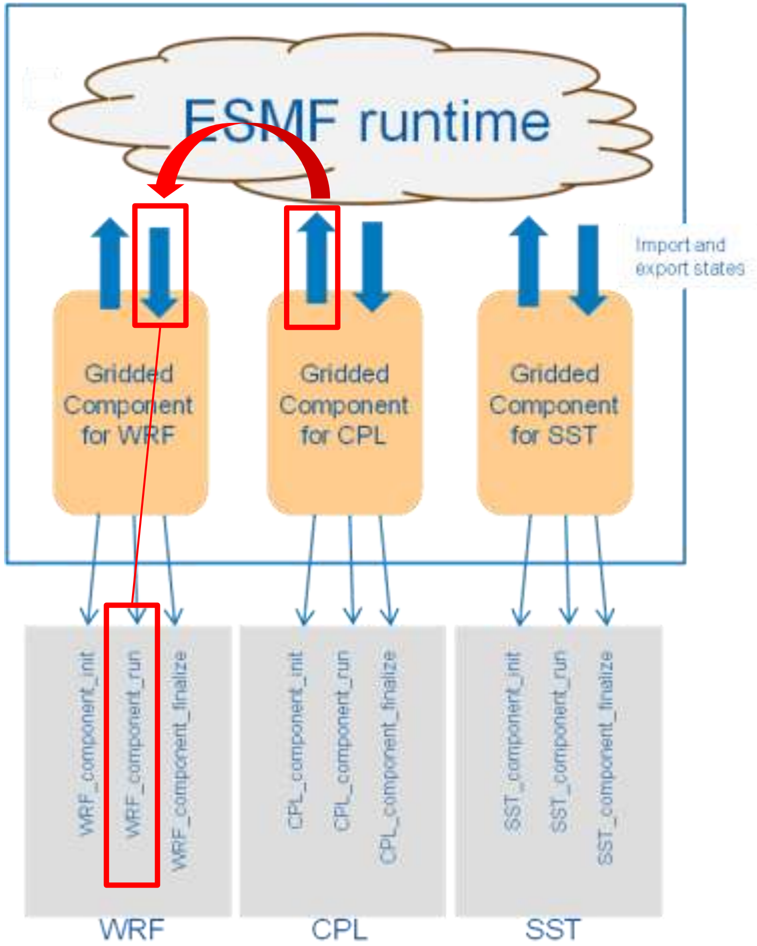
```
SUBROUTINE sst_component_run(…,importstate,exportstate,…)
   TYPE(ESMF_State) :: importState, exportState
   TYPE(ESMF_Field) :: field
   REAL(ESMF_KIND_R4), DIMENSION(ips:ipe,jps:jpe) :: inptr
   REAL(ESMF_KIND_R4), DIMENSION(:,:), POINTER    :: optr
   DO i = 1,datacount ! Number of fields
     read field from file into inptr
     CALL ESMF_StateGet( exportState,
                         TRIM(datanames(i)),
                         field )
     CALL ESMF_FieldGet( field, 0,
                         ptr )
     DO j = jps, jpe
       DO i = ips, ipe
         optr(i,j) = inptr(i,j)
       ENDDO
     ENDDO
   ENDDO
END SUBROUTINE sst_component_run
```

Case: WRF and ESMF




```
PROGRAM wrf_SST_ESMF
 CALL ESMF_Initialize( vm=vm, ...
 compWRF = ESMF_GridCompCreate(name="WRF", …
 compSST = ESMF_GridCompCreate(name="SST", …
 compCPL = ESMF_GridCompCreate(name="CPL", …
 importWRF = ESMF_StateCreate( …
 exportWRF = ESMF_StateCreate( …
 importSST = ESMF_StateCreate( …
 exportSST = ESMF_StateCreate( …
 importCPL = ESMF_StateCreate( …
 exportCPL = ESMF_StateCreate( …
 CALL ESMF_GridCompSetServices(compWRF, WRF_register,
 CALL ESMF_GridCompSetServices(compSST, SST_register,
 CALL ESMF_GridCompSetServices(compCPL, CPL_register,
 CALL ESMF_GridCompInitialize(compWRF,
        importStateWRF, exportStateWRF, clock, …
 CALL ESMF_GridCompInitialize(compSST,
        importStateSST, exportStateSST, clock, …
 CALL ESMF_GridCompInitialize(compCPL,
        importStateCPL, exportStateCPL, clock, …
 DO WHILE ( .NOT. ESMF_ClockIsStopTime(clock))
  CALL ESMF_GridCompRun(compSST,
        importStateSST, exportStateSST, clock, …
  CALL ESMF_CplCompRun(compCPL,
        exportStateSST, importStateWRF, clock, …
  CALL ESMF_GridCompRun(compWRF,
        importStateWRF, exportStateWRF, clock, …
 ENDDO
 CALL ESMF_GridCompFinalize(compSST, …
 CALL ESMF_GridCompFinalize(compWRF, …
 CALL ESMF_GridCompFinalize(compCPL, …
END PROGRAM wrf_SST_ESMF
```

Case: WRF and ESMF




```
PROGRAM wrf_SST_ESMF
 CALL ESMF_Initialize( vm=vm, ...
 compWRF = ESMF_GridCompCreate(name="WRF", …
 compSST = ESMF_GridCompCreate(name="SST", …
 compCPL = ESMF_GridCompCreate(name="CPL", …
 importWRF = ESMF_StateCreate( …
 exportWRF = ESMF_StateCreate( …
 importSST = ESMF_StateCreate( …
 exportSST = ESMF_StateCreate( …
 importCPL = ESMF_StateCreate( …
 exportCPL = ESMF_StateCreate( …
 CALL ESMF_GridCompSetServices(compWRF, WRF_register,
 CALL ESMF_GridCompSetServices(compSST, SST_register,
 CALL ESMF_GridCompSetServices(compCPL, CPL_register,
 CALL ESMF_GridCompInitialize(compWRF,
        importStateWRF, exportStateWRF, clock, …
 CALL ESMF_GridCompInitialize(compSST,
        importStateSST, exportStateSST, clock, …
 CALL ESMF_GridCompInitialize(compCPL,
        importStateCPL, exportStateCPL, clock, …
 DO WHILE ( .NOT. ESMF_ClockIsStopTime(clock))
  CALL ESMF_GridCompRun(compSST,
        importStateSST, exportStateSST, clock, …
  CALL ESMF_CplCompRun(compCPL,
        exportStateSST, importStateWRF, clock, …
  CALL ESMF_GridCompRun(compWRF,
        importStateWRF, exportStateWRF, clock, …
 ENDDO
 CALL ESMF_GridCompFinalize(compSST, …
 CALL ESMF_GridCompFinalize(compWRF, …
 CALL ESMF_GridCompFinalize(compCPL, …
END PROGRAM wrf_SST_ESMF
```

Case: WRF and ESMF

Configuring ESMF Coupling in WRF

- WRF uses I/O streams for ESMF coupling (Thanks Tom!)
- At compile time in the Registry
  - Export variables to ESMF by adding them to the variable set for an auxiliary output stream
  - Import variables from ESMF by adding them to the variable set for an auxiliary input stream
- At run time in the time_control section of the namelist.input file
  - Set the io_form=7 for the streams defined above
  - Set up any start, stop, and interval information
- Note:
  - Only one stream in and one stream out
  - Works only for one domain.

Case: WRF and ESMF

Compiling the example

- See: `./external/io_esmf/README.io_esmf`
- Set environment to point to ESMF 4.x on your system

```
setenv ESMFLIB=/mmm/users/michalak/esmf/lib/libg/AIX.default.64.mpi.default
setenv ESMFINC=/mmm/users/michalak/esmf/mod/modg/AIX.default.64.mpi.default
```

- Configure WRF

  `./configure` Then select the dmpar option

- Compile

```
./compile em_real
```

- Resulting executable: `main/wrf_SST_ESMF.exe`

Case: WRF and ESMF

Running the example

- Change directories to: `/test/em_esmf_exp/README_WRF_CPL_SST.txt`
- Refer to: `README_WRF_CPL_SST.txt` in that directory
- Unpack the coupler test configuration and data

```
% gunzip -c WRF_CPL_SST.tar.gz | tar xvf -
-rw-r--r--   6368 Feb 27 13:16 namelist.input.jan00.ESMFSST
-rw-r---r--   6368 Feb 27 13:16 namelist.input.jan00.NETCDFSST
-rw-r---r--   1286 Feb 27 14:51 real.csh
-rwxr-xr-x    948 Feb 27 14:51 real.lsf.csh
-rw-r---r-- 458064 Oct 12 11:58 sstin_d01_000000
-rw-r---r--   1074 Feb 27 14:51 test4_0.csh
-rwxr-xr-x    732 Feb 27 14:51 test4_0.lsf.csh
-rw-r---r--   1162 Feb 27 14:51 test4_0_ESMFSST.csh
-rw-r---r--    824 Feb 27 14:51 test4_0_ESMFSST.lsf.csh
-rw-r---r--   1190 Feb 27 14:52 test4_0_NETCDFSST_wrfexe.csh
-rw-r---r--    824 Feb 27 14:52 test4_0_NETCDFSST_wrfexe.lsf.csh
```

- Download the Jan 24, 2000 test case for WRF --- ask wrfhelp@ucar.edu
- Run the case (this is an LSF batch command script):

```
bsub < test4_0_ESMFSST.lsf.csh
```

- Save the files from the run in a subdirectory

```
mkdir test4_0_ESMFSST.out
$WRFDIR/test/em_esmf_exp >> mv PET?.ESMF* namelist.input rsl.*.* test4_0_ESMFSST.*.*
   wrfout* test4_0_ESMFSST.out
```

Case: WRF and ESMF

Running the example (continued)

- Rerun the case, this time just WRF, uncoupled

```
bsub < test4_0_NETCDFSST_wrfexe.lsf.csh
```

- Save the files from the standalone run in another subdirectory

```
% mkdir test4_0_NETCDFSST.out
% mv PET?.ESMF* namelist.input rsl.*.* test4_0_NETCDFSST.*.* wrfout*
   sstout_d01_000000 test4_0_NETCDFSST.out
```

- Compare the outputs from the two runs. Should be bit for bit

  % cmp -l test4_0_ESMFSST.out/wrfout_d01_2000-01-24_12:00:00 test4_0_NETCDFSST.out/wrfout_d01_2000-01-24_12:00:00 | wc

  0 0 0 (you should see three zeros like this)

- This is telling you that output from a run gets the data through ESMF from the coupled dummy ocean is exactly the same as ouput from a WRF run that reads the data itself. Ie. Coupling is correct and has no effect on the data.

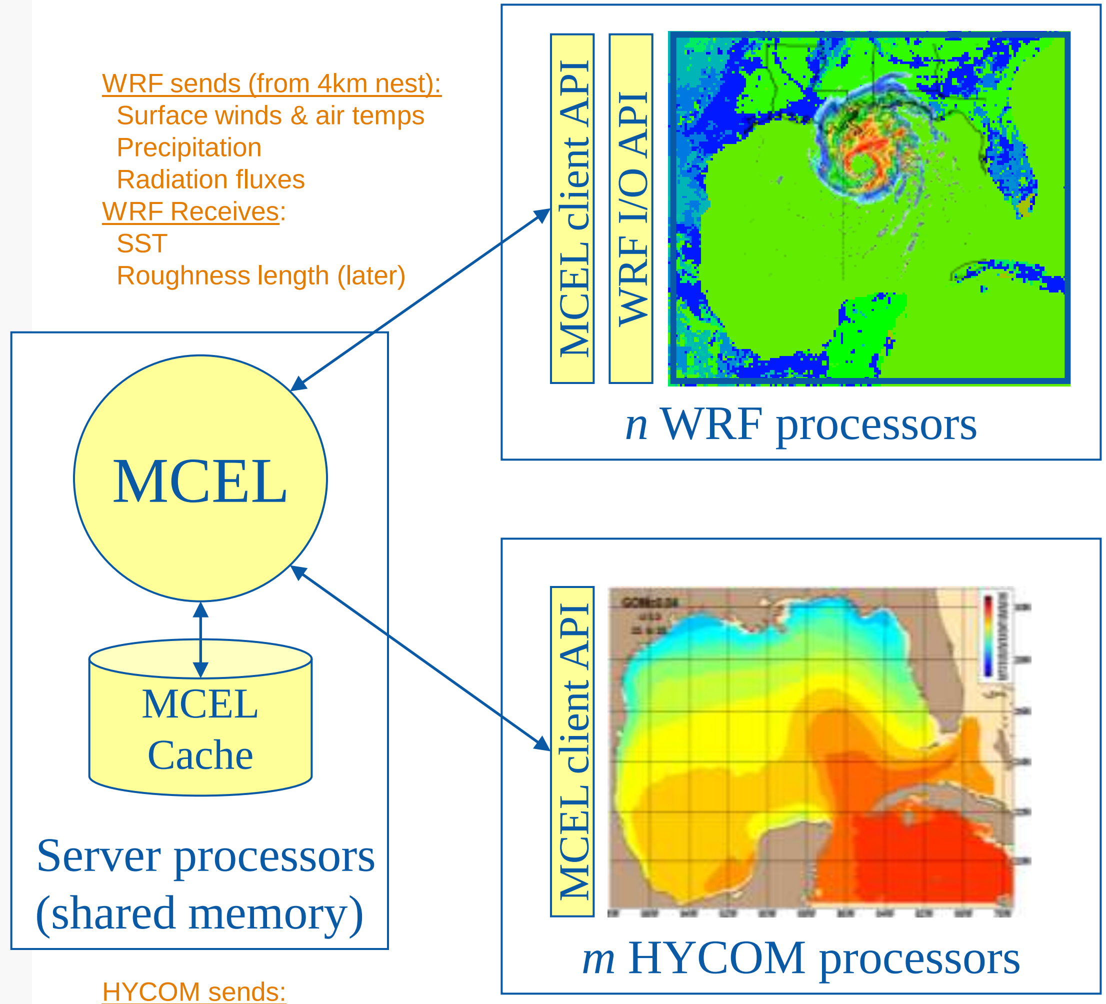
Case: WRF and ESMF

Summary

- Engineer applications into components
  - Modify top-level to add entry points
  - Write the code to initialize and define ESMF states
  - Wire them together
  - Write the top-level driver
  - Example is sequential coupled; concurrent is also possible
- Extending the example
  - Supporting different grids means adding interpolation or “regridding” in ESMF terminology
  - ESMF 4 now supports on-line regridding (non-conservative). Conservative regridding still uses off-line generated weights
  - Nesting and moving nests is not supported

Case: WRF and MCEL

- WRF and HYCOM couple through existing I/O modules
- Concurrent execution on different sets of processors
- Applications synchronize themselves based on data flow through MCEL
- Regridding support
  - Built-in to MCEL, transparent
  - Uses geo-location data already provided by applications
  - Supports any projection
  - Structured or unstructured grids
- MCEL server-managed data cache
  - Easy to switch between on-line and off-line coupling
  - Especially useful for WRF to generate spin-up data for HYCOM
  - Fault-tolerance
- Easy to add other models (e.g. wave) to configuration; simply another client



HYCOM sends:
SSTs
HYCOM Receives:
Winds
Precipitation
Radiation fluxes

***Model Coupling Environment Library, M. Bettencourt, AFRL**
HYCOM image courtesy A. Wallcraft, NRL
HYCOM/MCEL courtesy P. Fitzpatrick & N. Tran, MSU

Case: WRF and MCEL

Configuring MCEL Coupling in WRF

- WRF uses I/O streams for MCEL coupling
  - Developed with Matt Bettencourt under DOD PET project
- At compile time in the Registry
  - Send variables to MCEL server by adding them to the variable set for an auxiliary output stream
  - Get variables from MCEL server by adding them to the variable set for an auxiliary input stream
- At run time in the time_control section of the namelist.input file
  - Set the io_form=5 for the streams defined above
  - Set up any start, stop, and interval information
- Allows multiple streams and coupling to multiple domains. Moving nests not implemented.

Case: WRF and MCEL

Modifying code for MCEL coupling

- Codes using MCEL must be linked to the MCEL, CORBA, pthreads, and C++ versions of NetCDF libraries (Fortunately these are included in the MCEL distribution
- Calls to MCEL are placed in the I/O routines and toggled on and off with conditionals depending on whether running coupled
  - Sending data from HYCOM to MCEL

```
call getData(filID,"U10",aU10,start_time,end_time, &
  MCEL_TIMECENT_POINT,1,MCEL_FETCHPOLICY_BLOCK,ierr)
```

  - Getting data to HYCOM from MCEL

```
call storeData(progID,"SST",ahycomSST,start_time, &
  end_time,MCEL_TIMECENT_POINT,ierr)
```

  - Like ESMF, needs some setup, encapsulated in the HYCOM initialization routines

Case: WRF and MCEL

```
 if ( root ) then
        call newGrid(gridID,2,MCEL_GRIDTYPE_STRUCTURED,
     &              MCEL_GRIDCENT_NODAL,
     &              MCEL_GRIDCOORD_LATLONG,ierr)
        sizes(1) = ITDM
        sizes(2) = JTDM
        call setsize(gridID,sizes,ierr)
        call setlocationsxy(gridID,lon8,lat8,ierr)
        call setMask(gridID,amask,ierr)

C Register a new program
        call newProgram(progID,"HYCOM",ierr)
        call addvar(progID,"SST",MCEL_DATATYPE_DOUBLE,ierr)
        call setgrid(progID,gridID,ierr)
        call finalize(progID,ierr)

    . . .
```

Case: WRF and MCEL

Compiling

- Download and install MCEL
- Build application normally and link with addition of libraries.
  - MCEL
  - CORBA (openOrb)
  - Pthreads
  - C++ versions of NetCDF libraries
  - Fortunately these are included in the MCEL distribution

Case: WRF and MCEL

Running – at command line or in batch script

- Start up the MCEL server and interpolation program first.
  - These create .ior “magic cookie” files with TCP/IP information for the components to find them on the network
- Then start up the components of the coupled system
  - Components start up, initialize client side of MCEL
    - Look for .ior file and get cookie
    - Use cookie to open sockets to the MCEL server and interpolation program
    - Start running and then read and write coupling data over the TCP-IP sockets through the calls to getData and storeData
- Making sure everything starts up can be an issue for batch scheduled environments.
  - Run when there’s enough resources
  - Make friends with your system administrators

MCT Interpolation

Interpolation is treated as a sparse-matrix multiply: y = Mx

MCT datatypes:

- M SparseMatrix: hold distributed elements of the matrix (elements calculated offline using SCRIP)
- y, x AttributeVectors: distributed data type holding all data to be interpolated.
- x: data (T, q, u, v, etc.) on original grid and decomposition
- y: same data interpolated to new grid and decomposition.

Slides courtesy Rob Jacob, ANL

MCT Interpolation

X and Y are distributed over processors. How do you distribute M?

MCT lets you choose:

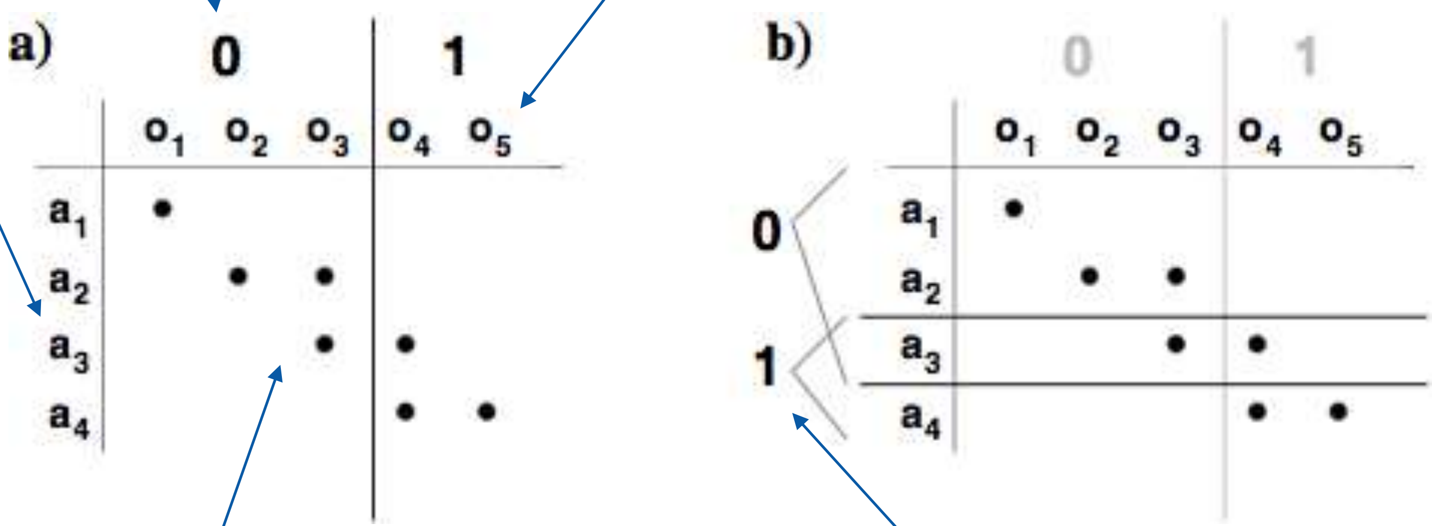
- Distribute elements according to distribution of Y
- Distribute elements according to distribution of X.
- MCT methods:
  - SparseMatrixComms,
  - SparseMatrixtoMaps (“Map” is MCT GlobalSegMap, a decomposition descriptor)

Matrix decomposition (according to y)

Ocean grid decomp.

Points on
atm grid (x)

Points on ocean grid (y)



Non-zero matrix elements
(interpolation weights)

Derived decomposition
For atmosphere (x'). Note:
some points needed more
than once

Interpolation steps

Initialization

- Read in matrix elements
- Distribute matrix elements in "y" order (SparseMatrixComms)
- Derive new decomposition descriptor for x, call it x' (SparseMatrixToMaps)
- Derive MCT Rearranger datatype to describe how to move data from x to x' (Rearranger)

Run

- Use MCT Rearrange method to move data from x to x'.
- Perform data-local multiply of elements of x' and M. Result is y

MCT MatAttrVectMul class can perform multiply and, if requested, the rearrange.

Summary

Coupling is non-trivial

- WRF supports interfaces to multiple coupling layers but it's a bigger problem than just WRF
- Be ready to invest time in reengineering codes
- Be ready to look beyond the plumbing
- Be ready to be surprised at the many new ways coupled modeling systems can fail or just be weird

Summary

Useful links

- ESMF: www.earthsystemmodeling.org
- MCEL: www.bettencourt.info/MCEL
- MCT: www.mcs.anl.gov/research/projects/mct
- WRF Software: www.mmm.ucar.edu/wrf/WG2/software_2.0

A few examples of coupled modeling systems:

- CCSM: *www.ccsm.ucar.edu*
- *GEOS-5: opensource.gsfc.nasa.gov/projects/GEOS-5/index.php*
- *NEMS: ams.confex.com/ams/pdfpapers/154223.pdf*