The Earth System Modeling Framework:

A high performance software solution for building and coupling model components.

Seminar at the Max Planck Institute for Meteorology, Hamburg

Gerhard Theurich, Bob Oehmke, Cecelia DeLuca
NOAA/CIRES

May 10, 2012

http://www.earthsystemmodeling.org/
Outline

- Motivation and History
- Architecture
- Regridding
- Interoperability (NUOPC)
- Wrap-Up
In climate research and numerical weather prediction...

... increased emphasis on detailed representation of individual physical processes; requires many teams of specialists to contribute components to an overall modeling system.

In computing technology...

... increase in hardware and software complexity in high-performance computing, as we shift toward the use of multi-paradigm parallel computing architectures.

In software...

... emergence of frameworks to promote code reuse and interoperability.

- ESMF is a focused community effort to tame the complexity of models and the computing environment.
- It leverages, unifies and extends existing software frameworks, creating new opportunities for scientific contribution and collaboration.
History

Phase 1: 2002-2005

NASA’s Earth Science Technology Office ran a solicitation to develop an Earth System Modeling Framework (ESMF).

A multi-agency collaboration (NASA/NSF/DOE/NOAA) won the award. The core development team was located at NCAR.

A prototype ESMF software package (version 2r) demonstrated feasibility.

Phase 2: 2005-2010

New sponsors included Department of Defense and NOAA.

Many new applications and requirements were brought into the project, motivating a complete redesign of framework data structures (version 3r).

Phase 3: 2010-2015

The core development team moved to NOAA/CIRES for closer alignment with federal models.

Basic framework development has been completed with version 5r (ports, bugs, feature requests, user support etc. still require resources).

Extensions continue: regridding, interoperability and language bindings.

The focus is on increasing adoption and creating a community of interoperable codes.
Backward Compatibility

- Starting with ESMF v5.2.0r, > 75% of the API are marked as backward compatible.
- User code relying on these calls will compile with future versions of ESMF (unchanged!).
- Provides a solid platform for application development.
- Some newer interfaces are exempt (e.g. location streams, exchange grids, ...).
- Keyword enforcement (e.g., rc=localrc) for optional arguments is an important mechanism.
- Additional optional arguments can be introduced.
- Additional methods or overloads can be introduced.
Outline

- Motivation and History
- Architecture
- Regridding
- Interoperability (NUOPC)
- Wrap-Up
• Base library of about 500,000 lines of source code.
• 60% in Fortran, 40% in C/C++
• Complete Fortran API:
  - use ESMF
  - Derived types and methods
• Limited C API:
  - #include "ESMC.h"
  - Structs and methods
• Emerging Python API
  - import ESMP
  - Classes with methods
- Unix/Linux and Windows (Cygwin/MinGW) systems
- Based on MPI (bypass mode “mpiuni” as option)
- OpenMP and Pthreads support
- I/O through NetCDF/HDF, Xerces, or PIO
- Sockets for web services and fault-tolerance extensions
- Highly portable: tested on > 40 different OS/Compiler/MPI combinations every night.
- Over 170,000 lines of example, unit and system testing code.
Infrastructure

Architecture – Utility Classes

- Config
- Log

Diagram:
- BaseTime
- TimeInterval
- Clock
- Time
- Calendar
- Alarm

Relations:
- TimeInterval to Clock: 1
- Clock to TimeInterval: 0..n
- TimeInterval to Alarm: 2
- Alarm to TimeInterval: 0..n
- Clock to Time: 0..n
- Time to Clock: 5
- Time to Calendar: 0..n
- Calendar to Time: 1
- Alarm to Calendar: 0..n
- Alarm to Time: 0..n

TimeMgr
Architecture – Components and States

Superstructure

GridComp

User Code

VM

Infrastructure

State Import:

State Export:

Initialize
Run
Finalize
WriteRestart
ReadRestart

User Code
Components can be arranged hierarchically, helping to organize the structure of complex models.

Different modeling groups may create different kinds or levels of components.
Architecture – Component Overhead

- Overhead of ESMF component wrapper around native CCSM4 component.
- For this exercise, ESMF wrapping required **NO code changes** to the scientific modules.

- No significant performance overhead (<3% is typical)
- Few code changes for codes that are modular.

- Platform: IBM Power 575, bluefire, at NCAR
- Versions: CCSM_4_0_0_beta42 and ESMF_5_0_0_beta_snapshot_01
- Resolution: 1.25 degree x 0.9 degree global with 17 vertical levels for both the atmospheric and land model. Ocean model resolution is 320x384x60.
### Architecture – Distributed Data I

#### Regular block decomposition

<table>
<thead>
<tr>
<th>A11</th>
<th>A12</th>
<th>A13</th>
</tr>
</thead>
<tbody>
<tr>
<td>A21</td>
<td>A22</td>
<td>A23</td>
</tr>
<tr>
<td>A31</td>
<td>A32</td>
<td>A33</td>
</tr>
<tr>
<td>A41</td>
<td>A42</td>
<td>A43</td>
</tr>
<tr>
<td>A51</td>
<td>A52</td>
<td>A53</td>
</tr>
<tr>
<td>A61</td>
<td>A62</td>
<td>A63</td>
</tr>
</tbody>
</table>

#### Irregular block decomposition

<table>
<thead>
<tr>
<th>A11</th>
<th>A12</th>
<th>A13</th>
</tr>
</thead>
<tbody>
<tr>
<td>A21</td>
<td>A22</td>
<td>A23</td>
</tr>
<tr>
<td>A31</td>
<td>A32</td>
<td>A33</td>
</tr>
<tr>
<td>A41</td>
<td>A42</td>
<td>A43</td>
</tr>
<tr>
<td>A51</td>
<td>A52</td>
<td>A53</td>
</tr>
<tr>
<td>A61</td>
<td>A62</td>
<td>A63</td>
</tr>
</tbody>
</table>

#### Arbitrary decomposition

- A52 A13 A11 A62
- A21 A53 A61
- A43 A33 A51 A32 A22
- A23 A42 A31 A41

- Local memory allocations remain indexed in 2D.
- Grid has a more restrictive irregular block decomposition.
- Undistributed dimensions add to allocation dimensionality.

- Local memory allocations 1D.
- Arbitrarily decomposed logically rectangular data.
- Unstructured data.
- Undistributed dimensions add to allocation dimensionality.
Architecture – Distributed Data II

Array/DistGrid

DELayout

OS: processes + threads
- virtual address spaces (VASs)
- persistent execution threads (PETs)

Hardware:
- Proc. Elements (PEs)
- Compute units
- Single System Images

VM

PET 0
PET 1
PET 2

DE0
DE1
DE2
DE3
Field = Array + (some form of grid object)

- **Grid**
  - Structured representation of a region.
  - Logically rectangular coordinates in 1D, 2D, 3D.

- **LocStream**
  - Locations of a set of data points.
  - Any dimension, coordinates are stored as “keys”.

- **Mesh**
  - Unstructured representation of a region.
  - Nodes with 2D or 3D coordinates.
  - Elements: triangles/quadrilaterals for 2D parametric space
tetrahedrons/hexahedrons for 3D parametric space

- **Grid**
  - 2D boundary layer between two model grids.
  - Represented by a custom constructed Mesh.
• Motivation and History
• Architecture
• **Regridding**
• Interoperability (NUOPC)
• Wrap-Up
Regridding (or remapping or interpolation) is the process of moving data from one grid to another while preserving qualities of the original data.

- **Regrid Weight Generation**
  - Wide range of grids: structured and unstructured, global and regional, 2D and 3D.
  - Parallel, scalable and fast.

- **Sparse Matrix Multiplication**
  - Index space operation, independent of higher level interpretation.
  - Parallel, scalable and fast.
Regridding – Weight Generation Features

- **Flexible**
  - Computes weights between a wide range of grids: structured and unstructured, global and regional, 2D and 3D
  - Options for interpolation method, pole treatment, masked points, ...

- **Accurate and Portable**
  - Spherical regridding handled in 3D to avoid pole issues
  - Tested nightly on many platforms

- **Parallel and Fast**
  - Able to compute weights in minutes which before took hours
  - Able to compute weights between very large grids

- **Community developed**
  - Supported by NASA, NOAA, DOD and NSF funding
  - Well established (since 2005) community processes for prioritization, support and review.
  - Development priorities set by users through quarterly Change Review Board (CRB) meetings
Regridding – Weight Generation Features II

- Interpolation types
  - Bilinear
  - Higher order
    - Yields better derivatives/smooth results than bilinear.
    - Based on “patch recovery” used in finite element modeling [1][2].
  - First order conservative

- Masking
  - Source
  - Destination

- Unmapped destination point handling
  - Error
  - Ignore

- Pole options for global spherical logically rectangular Grids
  - Full circle average: artificial pole is average of all source points next to pole
  - N-point average: artificial pole is average of n top source neighbors of dest point
  - Teeth: gap at pole filled by triangles
  - No pole: error if destination point lies above top row of source points
Regridding – Weight Generation User Interfaces

- **Stand alone application (**offline**)**
  - Generates a NetCDF weight file from two NetCDF grid files.
  - Formats (all NetCDF): SCRIP, ESMF unstructured format, CF GridSpec structured convention, and CF UGrid unstructured convention.
  - Builds and installs with the ESMF source code distribution.

  ```
  mpirun -np 32 ESMF_RegridWeightGen -s src_grid.nc -d \
  dst_grid.nc -m bilinear -w weights.nc
  ```

- **API calls (**integrated**)**
  - ESMF library calls directly accessible during model run.
  - Interfaces available in Fortran, C and Python.
  - Access weights or directly store parallel sparse matrix multiply communication.
  - Can be used without other parts of ESMF (e.g. components are not needed).

  ```
  call ESMF_FieldRegridStore(srcField=src, dstField=dst, &
  regridMethod=ESMF_REGRID_METHOD_BILINEAR, routehandle=rh)
  ```

  ```
  call ESMF_FieldRegrid(srcField=src, dstField=dst, &
  routehandle=rh)
  ```
Regridding – Offline supported grids

- Grids with spherical (lon, lat) coordinates.
- Mix and match pairs of:
  - Global 2D logically rectangular grids
  - Regional 2D logically rectangular grids
  - 2D unstructured meshes composed of polygons with any number of sides (triangles, quadrilaterals, pentagons, hexagons):
    - ESMF internally represents these as triangles and quadrilaterals
- Multi-patch grids (e.g. cubed spheres) currently supported via unstructured formats.
- Multi-patch support expected with complete GridSpec implementation.
- 3D Cartesian unstructured grids.
Regridding – Integrated supported grids

- In addition, integrated regridding supports Cartesian (x,y) coordinates:
  - Regridding between any pair of:
    - 2D meshes composed of triangles and quadrilaterals
    - 2D logically rectangular grids composed of a single patch

- Bilinear or conservative regridding between any pair of:
  - 3D meshes composed of hexahedrons
  - 3D logically rectangular grids composed of a single patch
Regridding – Weight Generation Performance

- Always go through unstructured Mesh.
- Increases flexibility.
- Small add. overhead to bilinear interpolation.
- Greatly improves performance over existing conservative methods.

- Previous solution takes 635s (20x) to compute conservative weights
- Previous solution unable to compute bilinear weights from cubed sphere

- Platform: Crag XT4, jaguar at ORNL
- Version: ESMF_5_2_0_beta_snapshot_07
- fv0.47x0.63: CAM Finite Volume grid, 576x384
- ne60np4: 0.5 degree cubed sphere grid with pentagons, 180x180x6
Regridding – Weight Generation Impacts I

- Higher order interpolation leads to reduced noise in wind stress values
  - User: Community Earth System Model
  - Grids: CAM atmosphere lat/lon to POP ocean displaced pole lat/lon
  - Impact: ESMF patch interpolation reduced imprint of coarser resolution atmosphere grid on ocean for interpolated wind stress values. Interpolation weights used in CCSM4 and subsequent IPCC runs

- Better interpolation of cubed sphere (unstructured) and lat/lon ocean
  - User: Community Earth System Model
  - Grids: HOMME cubed sphere atmosphere to lat/lon ocean grid
  - Impact: ESMF conservative regridding enabled easier integration of a high resolution dynamical core into CAM, reduced distortion near the pole.

- Enables CLM land model to run on cubed sphere
  - User: Community Earth System Model
  - Grids: Land lat/lon to HOMME cubed sphere
  - Impact: ESMF parallel bilinear mapping from lat/lon to HOMME cubed sphere allowed investigation of high resolution land model to move forward for CESM.

- Better values at poles for unstructured to lat/lon remapping
  - User: Community Earth System Model
  - Grids: NCAR MPAS unstructured grid to POP ocean grid
  - Impact: ESMF conservative interpolation solved problems with negative weights at the pole.
Regridding – Weight Generation Impacts II

- Allows fast interpolation of enormous topography data set
  - User: NASA Global Modeling and Assimilation Office
  - Grids: 4km global lat/lon grid to 7km cubed sphere grid
  - Impact: ESMF conservative regridding allows the interpolation of topography data in 1.5 minutes, which otherwise would take hours.

- Provides ability to do fast parallel interpolation between geographic and magnetic grids
  - User: NCAR High Altitude Observatory
  - Grids: Global magnetic grid and global geographic grid
  - Impact: ESMF bilinear regridding provides fast parallel interpolation to allow interpolation between two different grids with two different distributions during run of Thermosphere Ionosphere Mesosphere General Circulation Model (TIME-GCM).

- Allows fast interpolation of data between very large meshes
  - User: Community Surface Dynamics Modeling System
  - Grids: 16 million triangle mesh to 16 million triangle mesh
  - Impact: ESMF bilinear regridding allows interpolation of data between two large meshes that other packages could not handle for the surface dynamics community.
Regridding – Sparse Matrix Multiplication Intro

- Method

\[ \text{ESMF\_FieldRegridStore}(..., \text{routehandle}=\text{rh}) \]

actually does this:

- Compute interpolation weights for src → dst regridding
- Call \[ \text{ESMF\_FieldSMMStore}(..., \text{routehandle}=\text{rc}) \]

- Method

\[ \text{ESMF\_FieldRegrid}(..., \text{routehandle}=\text{rh}) \]

actually does this:

- Call \[ \text{ESMF\_FieldSMM}(..., \text{routehandle}=\text{rc}) \]
Regridding – Sparse Matrix Multiplication Features

- Apply coefficients (weights) in parallel to distributed data.
- Index space operation, independent of higher level interpretation (Regrid, Redist, Halo).
- Scalable parallel store implementation based on distributed directory. [3]
- Auto-tuning during store for optimal execution (outstanding comms, src/dst balance).
- Parallel execution, overlapping communication with computation.
- Options to initialize destination elements: total, select, empty.
- Non-blocking execution option, with test and wait.
Regridding – Sparse Matrix Multiplication Performance

- Community Climate System Model (CCSM) atmosphere to ocean grid remapping.
- Native data structures wrapped in ESMF Arrays with data referencing.

- Comparable performance to native code, slightly better scaling at higher processor counts

- Versions: ESMF: 400rp2, CCSM: ccsm4_0_rel08
- Resolution: f05_t12 (fv 0.47x0.63 atmosphere/land, tripole 0.1 ocean or 576x384 atmosphere/land and 3600x2400 ocean)
Prototype!!!

Separate CVS download

Requirements
- python
- numpy
- ctypes

Limited platform support
- Linux/Darwin, GCC(g++/gfortran), OpenMPI.

Data type: ESMP_Field

Grid types:
- Single-tile 2D logically rectangular type: ESMP_Grid
- Unstructured type: ESMP_Mesh

Support for all ESMF interpolation options.
import ESMP

ESMP.ESMP_Initialize()

field = ESMP.ESMP_FieldCreateGrid(grid, name)

routehandle = 
ESMP.ESMP_FieldRegridStore(srcfield, dstfield, 
srcMaskValues=_NP.array([1], dtype=_NP.int32), 
dstMaskValues=_NP.array([1], dtype=_NP.int32), 
regridmethod=ESMP.ESMP_REGRIDMETHOD_CONSERVE, 
unmappededaction=ESMP.ESMP_UNMAPPEDACTION_ERROR, 
srcFracField=srcfracfield, 
dstFracField=dstfracfield)

ESMP.ESMP_FieldRegrid(srcfield, dstfield, routehandle)
Outline

- Motivation and History
- Architecture
- Regridding
- Interoperability (NUOPC)
- Wrap-Up
Interoperability – ESMF

Coupling Options

- Fortran or C components
- Single executable
- Multiple executable
  - Web service option
  - Top level MPMD
  - Fault-tolerant components (new in 5.3.0)
- Coupling communications can be called either from within a coupler or directly
  - from a gridded component – useful when it is inconvenient to return from a component in order to perform a coupling operation
- Recursive components for nesting higher resolution regions
- Ensemble management with either concurrent or sequential execution of ensemble members
Interoperability – Community of Components

ESMF Model Map 2010
Interoperability – Components, a first step

- Interoperability on API level
  - Component interaction through: ESMF_State + ESMF_Clock
  - Independently developed codes will compile and link.

- Interoperability problem pushed into run-time

- Small set of ESMF data structures in State
  - ESMF_Field, ESMF_Array, ESMF_FieldBundle, ESMF_ArrayBundle, ESMF_RouteHandle, ESMF_State
  - Hold enough information for basic compatibility checking

- Need a common model architecture
- Need more metadata for checks and self doc
Interoperability - Attributes

- Attributes hold meta data in name/value pairs.
  - `AttributeAdd() / Set() / Get()`

- Supported for:
  - `GridComp/CplComp`
  - `State`
  - `Field/FieldBundle`
  - `Grid`
  - `Array/ArrayBundle`
  - `DistGrid`

- Attribute hierarchies for model documentation.
- Attribute packages for standards:
  - `CF, ISO, METAFOR/CIM`
- XML output option.
Interoperability - NUOPC

- National Unified Operational Prediction Capability
  - Consortium of U.S. operational weather and water prediction centers.
- Standardize the use of ESMF across NOAA, Navy, Air Force, NASA, and other model applications.
- Demonstrate improved level of interoperability.
- Develop a Common Model Architecture (CMA).
- NUOPC websites
  - http://www.weather.gov/nuopc
  - http://www.earthsystemmodeling.org/conventions/nuopc.shtml
Interoperability – NUOPC CMA Products

• “NUOPC Layer” as “add-on” in the ESMF distribution
  ➔ Generic Components:
    Pre-compiled generic code with well-defined specialization points.
  ➔ Utility Routines:
    E.g. NUOPC_FieldBundleUpdateTime(), NUOPC_PrintCurrTime(), ...
  ➔ Metadata dictionaries:
    Field dictionary with StandardName and CanonicalUnits.

• Example codes, serving as templates for users
  ➔ Explicit ATM/OCN coupling
  ➔ Explicit ATM/OCN with mediator coupling
  ➔ Simple Implicit ATM/OCN
  ➔ Implicit ATM/OCN coupling with two time levels
  ➔ And more...

• Compliance Checker as runtime option.
• Regular telecons, trackers, documents.
Interoperability – NUOPC CMA Details

- Major components (ATM, OCN, LND, ICE, WAV,...) are siblings.
- Major components are implemented as Gridded Components (ESMF_GridComp).
- Gridded Components own their Import State and Export State.
- Always two Gridded Components are connected through a Coupler Component (ESMF_CplComp).
- Components are not allowed to modify the incoming (driver) Clock.
- Components must synchronize their internal Clock against the incoming Clock.
- Components must implement compatibility checking (“current time” and “time step”) between internal and incoming Clock.
- Data between Gridded Components is exchanged through Fields (ESMF_Field).
- Fields used for data exchanged between Gridded Components must carry a minimum of metadata (conveniently supplied through the NUOPC Field Dictionary).
- Components must time stamp their export Fields.
- Components must check compatibility of the import Fields' time stamps.
Interoperability – NUOPC
Generic Components (GC)

- Fortran modules with generic `ESMF_GridComp` and `ESMF_CplComp` implementations.
- NUOPC Types: Model, Mediator, Driver, Connector.
Interoperability – NUOPC
GC: Public Elements

Fortran module:

```fortran
use NUOPC_Model, only: &

model_routine_SS => routine_SetServices, & → routines
model_routine_Run => routine_Run, &

model_type_IS => type_InternalState, & → derived types

model_label_IS => label_InternalState, & → labels
model_label_SetClock => label_SetClock, &
model_label_Advance => label_Advance, &
model_label_SetRunClock => label_SetRunClock, &
model_label_CheckImport => label_CheckImport
```
Interoperability – NUOPC
GC: Specialization

Standard Component Methods
(I/R/F/WR/RR)

Phases

Optional/required attachable method:
MethodSet(), e.g. for label_SetClock

Entry point for method + phase: SetEntryPoint()

Set a generic routine as entry point, e.g. routine_Run

Overwrite a generic attachable method with a phase specific one

Set some of the members of type_InternalState

InternalState

varA
varB
...

Phases

Phase 0

Phase 1

Phase 1

Phase 2

Phase 0
Interoperability – NUOPC CMA Adoption

- **Navy**: NOGAPS-HYCOM and NavGEM-HYCOM
  - Fully functional explicit and semi-implicit ATM-OCN coupling based on NUOPC Layer.

- **Navy**: COAMPS (NCOM, SWAN, WaveWatchIII)
  - Working prototype ATM-OCN-WAV coupling based on NUOPC Layer.

- **NOAA**: NEMS (GFS-HYCOM)
  - Prototype for ATM-OCN coupling based on NUOPC Layer under development.
Outline

- Motivation and History
- Architecture
- Regridding
- Interoperability (NUOPC)
- Wrap-Up
ESMF v5.3.0 just released!

- 5.3.0 is backward compatible with 5.2.0r-series
  - User code that works with 5.2.0r (or 5.2.0rp1, or 5.2.0rp2) will also work with 5.3.0.

- Regridding
  - 3D first order conservative regridding.
  - `ESMF_RegridWeightGen` supports GRIDSPEC (CF structured grid convention).
  - `ESMF_RegridWeightGen` supports UGRID (CF unstructured grid convention).
  - Conservative regridding supports user supplied areas.
  - Conservative regridding supports masking of unstructured Mesh.

- Prototype Python Interface

- Prototype fault-tolerant components

- NUOPC Layer
  - `NUOPC_Driver` supports petList for Models, Mediators and Connectors, allowing concurrent execution.
  - `NUOPC_Driver` and `NUOPC_Model` components support simple explicit, semi-implicit, as well as complex implicit component run sequences.

- XGrid creation supports lists of grids on either side (e.g. OCN+LND).
Active Development

- Python API
- NUOPC Layer
- Standard connectors to web services and other interface standards (e.g. OpenMI).
- Regridding options
- Fault-tolerance
- I/O (consistent, efficient, parallel)
- String based access to object information through Attributes
- Performance
Where to get HELP?

Documentation and training materials

● Users Guide, comprehensive Reference Manuals
● Many examples and system tests
● External demos, treating ESMF as external dependency - the way a user application would.

If you’re stuck

● Write the support line,

  esmf_support@list.woc.noaa.gov

If you’re really stuck

● We can usually arrange a call!
References

