# Porting the MIT Marine Ecosystem Model into CESM

Nancy J. Norton, NCAR

In collaboration with

Stephanie Dutkiewicz and Oliver Jahn (MIT)
Keith Lindsay, Gokhan Danabasoglu, and Matthew Long (NCAR)

# Where to Start?

- Project Planning Infrastructure
  - Project planning tools, create preliminary documentation, put documentation under revision control, plan documentation and code repository layouts
- Discovery Phase:
  - Darwin is not a stand-alone model; it is an optional “package” available for use in the MITgcm
  - Download MITgcm and Darwin code
  - Review code
  - Review MITgcm configuration and build methods
- Code Analysis
  - Analyze general structure of Darwin
  - Map out MITgcm infrastructure dependencies in Darwin (grids, I/O, communications, time/calendar)

# Getting Started

- Code Analysis Findings
  - Differences:
    - FORTRAN77 vs Fortran90
    - #include COMMON blocks vs "use module"
    - extensive use of Cpp options in MITgcm and Darwin; minimal use in POP2
    - Arakawa C vs B grids
  - Darwin tightly connected to MITgcm:
    - common variables
    - I/O
    - domain decomposition, communications and global operations (MPI)
    - grids
    - initialization
    - timestepping
    - clock/calendar

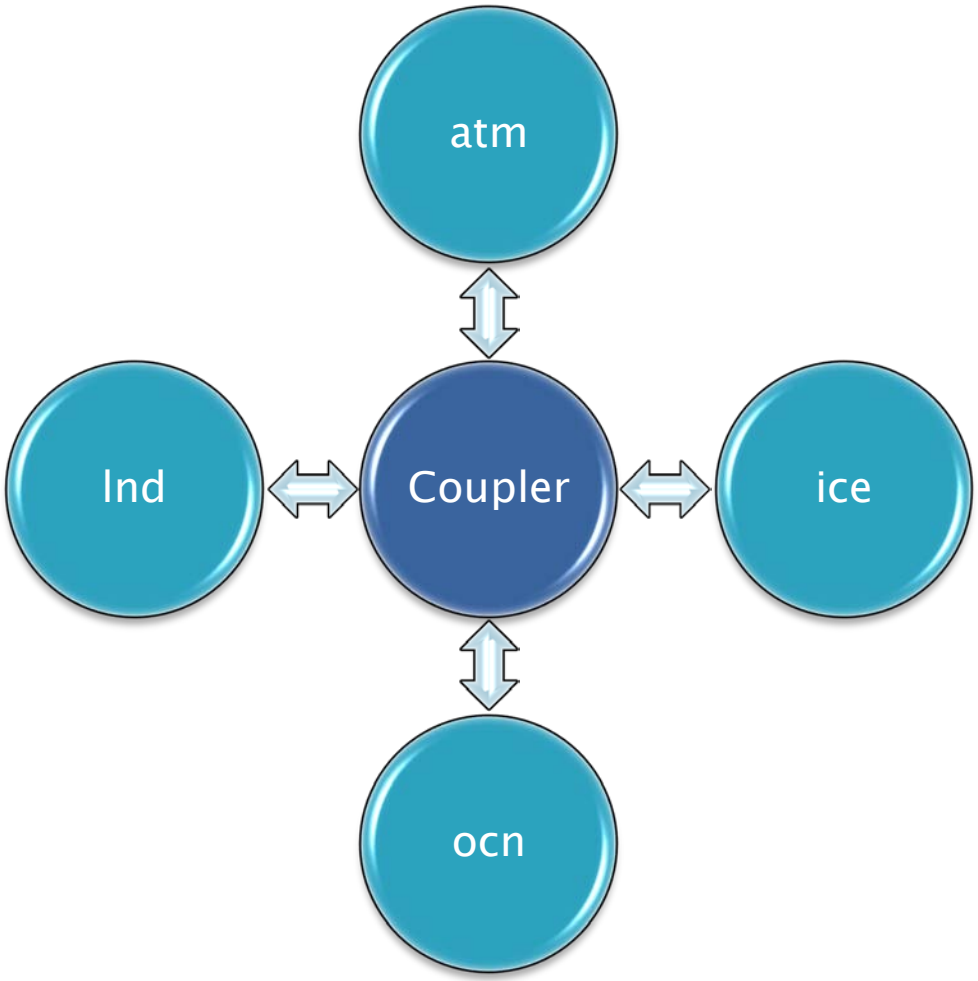
# Design Decision

- Rewrite Darwin as a CESM POP2 module (“pop-ify”), or use Darwin as-is, in a library accessible from CESM POP2?
  - From an implementation viewpoint, each approach had roughly equal appeal:
    - Risks
    - Benefits
    - Technical challenges
  - But, looking to the future, replacing Darwin with Quota would be essentially "free" with the Darwin-as-a-library approach.
  - Chose the Darwin-as-a-library approach

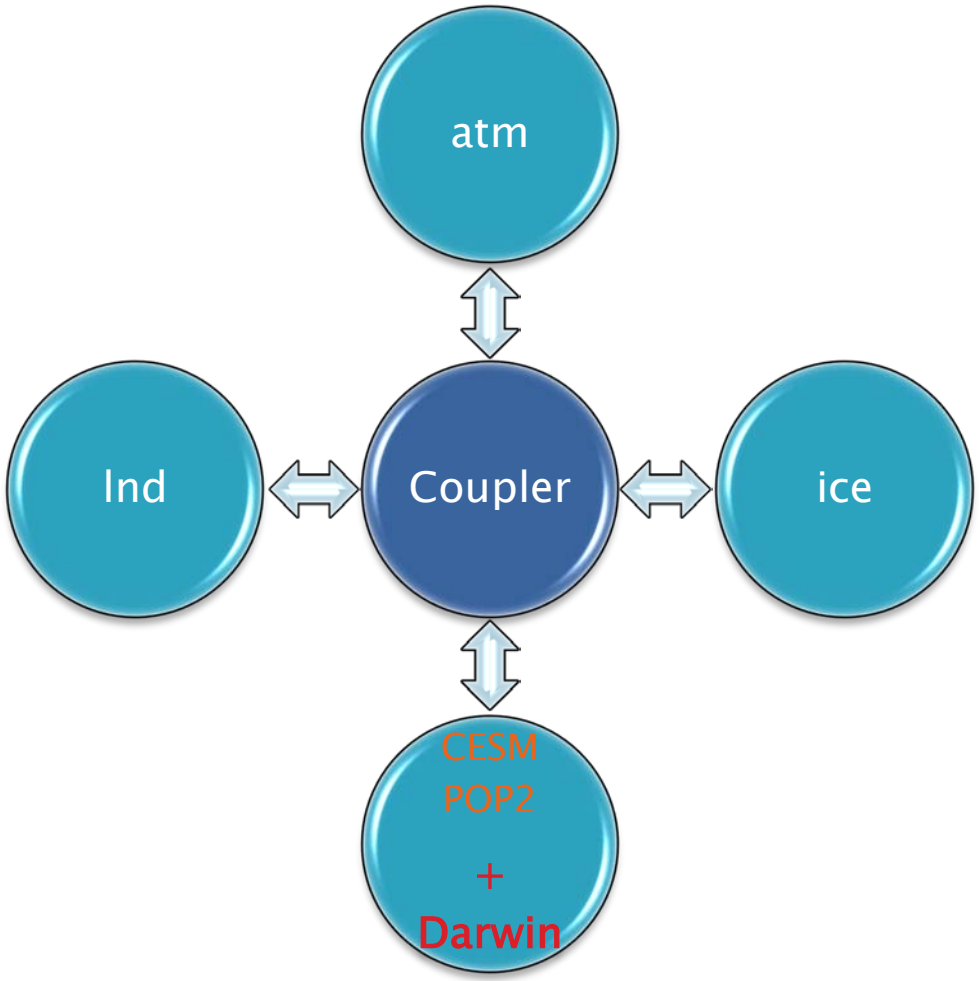
# So Now What?

- Develop a “mental model” of how to connect Darwin to CESM POP2
- Develop a simple proof-of-concept prototype
- If successful,  gradually develop full functionality
- Develop + Test
- But first, a short detour…

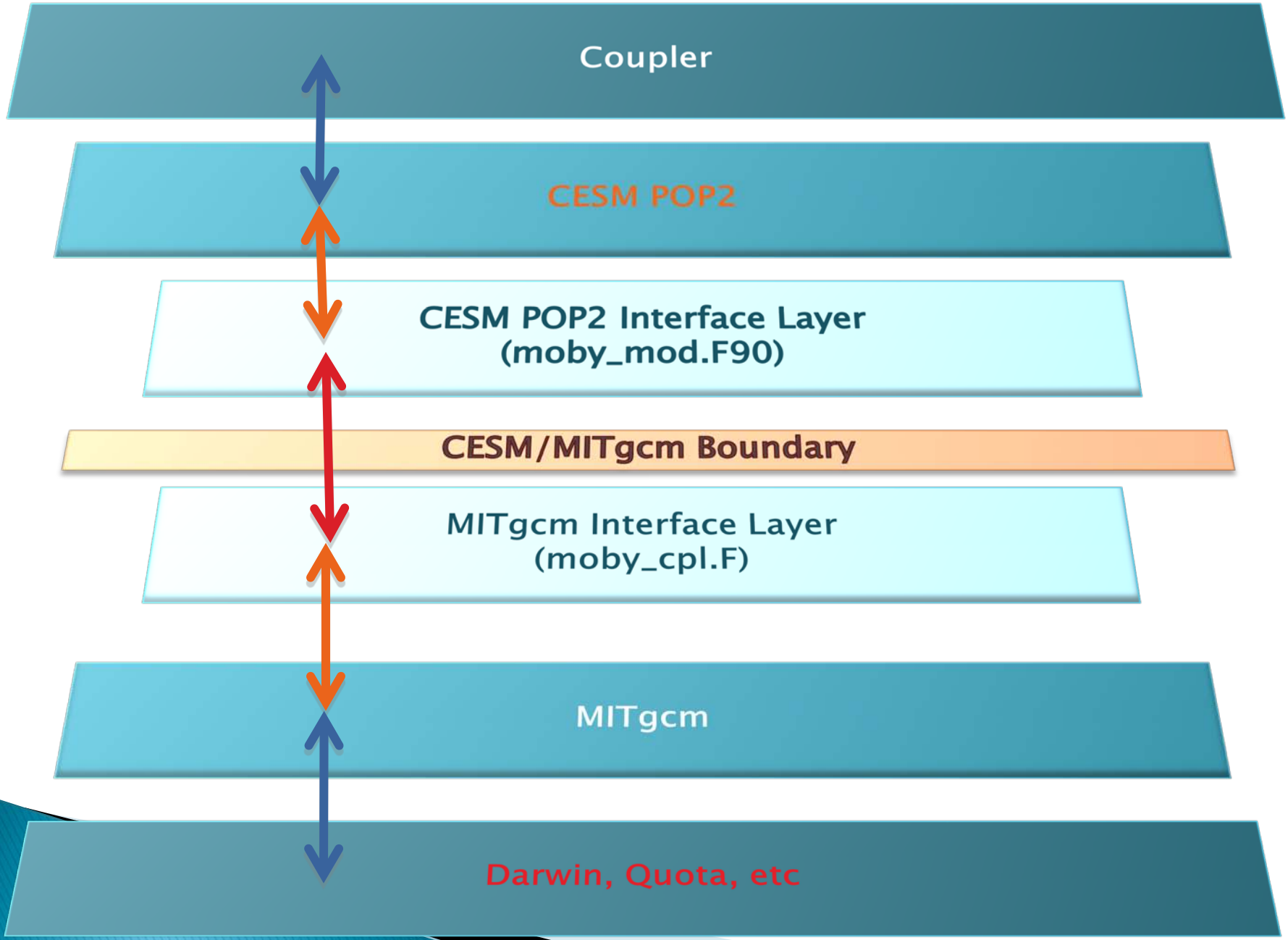
# Simplest Conceptual Model of CESM



# CESM with Active Ocean + DARWIN



# CESM POP/ MITgcm Marine Ecosystem Interface Diagram



# Prototype Preparation

- Establish Design Ground Rules
  - Use CESM infrastructure for case setup, configuration, build, and execution
  - Use CESM POP2 I/O, MPI communications, tracer support, grids, masking, and time-stepping
  - As much as possible, use MITgcm code "as-is," but it's ok to modify MITgcm routines and "include files" (bypass certain parts, eg)
  - All CESM MOBY modifications controlled by a single Cpp option, CESMMOBY
  - *Minimize mods to Darwin* -- will make replacing Darwin with Quota easier

# Prototype Development

- Feasibility Exploration/Simple Prototype Development
  - Set up a standard CESM test case
  - Modify CESM scripts to assemble Darwin code, build and link Darwin library, and run code  (challenge: moby.cpl7.template script)
  - Create simplest interface routines to connect POP and Darwin
  - “Hello world" success!
  - Shared MPI communicator success! (one, then four ocean processors)
- Have confidence in approach. Now Build up Interface Routines

# From Prototype to Working Model

- POP2 Interface: moby_mod.F90
  - Based on Keith Lindsay's ecosys_mod.F90
  - Uses Keith's passive-tracer support
  - Two classes of routines:
    - "moby_" routines hook into the CESM POP2 model in the same way as all other CESM POP2 passive-tracer routines.
    - "POP_moby" routines are the interface routines that reach across to the Darwin side to drive the Darwin model and to exchange information.

# moby_mod.F90

**Initialization**

- moby_init
- moby_init_tavg
- moby_init_sflux
- moby_init_interior_restore
- POP_mobyInit1
- POP_mobyInit2

**Science**

- moby_set_interior_3D
- moby_reset
- moby_set_sflux
- moby_tavg_forcing
- moby_global_tracer_volume
- extract_surf_avg
- comp_surf_avg
- POP_mobySurfaceForcingSet
- POP_mobyMeanArea
- POP_mobyCons

**Information exchange**

- POP_mobySendTime

**Misc**

- POP_mobyFinal
- POP_mobyConsistencyChecks

**I/O**

- moby_write_restart

# From Prototype to Working Model

- MITgcm/Darwin Interface: moby_cpl.F
  - Multiple subroutines in one file plus two include files
    - moby_cpl.F
    - CESM_CPL_PARAMS.h  (CESM "shared constants")
    - CESM_EEPARAMS.h  (CESM I/O; MPI support)
  - moby_cpl routines are called from moby_mod.F90 routines
  - Serve as drivers for MITgcm and Darwin routines
  - Exchange information (note: CESM POP2-centric "put" and "get")

# moby_cpl.F

## Initialization

moby_cplComm
moby_cplInitLog
moby_cplShrConstants
moby_cplInitializeFixed
moby_cplInitializeVaria
moby_cplIniGrid
moby_cplIniThreadingEnv

## Science

moby_cpl_call_Darwin_forcing
moby_cpl_call_Darwin_fe_chem
moby_cplSurfaceForcingSet
moby_cplSurfaceForcingResetFlags

## Information exchange

moby_cplTime
moby_cplWRAPPER (eeboot, the_model_main)
moby_cplGetNumPTRACERS
moby_cplGetInfoPTRACERS
moby_cplGetInfoPtrIndices
moby_cplGetInfoRatios
moby_cplGetInfoOptions
moby_cplGetPtracer
moby_cplPutPtracer
moby_cplPutTS
moby_cplPutQSW
moby_cplPut_hFacC
moby_cplPutGSM
moby_cplPutScalars

## Misc

moby_cplDocPtracers
moby_cplConsistencyChecks
moby_cplFinal
moby_cplFlush

# Testing

- Software Engineering Tests
  - Exact restart
  - CESM DEBUG initializes all model fields to NaNs, traps underflows, overflows, and illegal operations, and activates bounds checking.
  - Stability (1 year; long-term problems?)
  - Memory leak (no memory growth)
  - Memory scaling (with more processors, memory should scale accordingly)
  - Domain decomposition
  - Timing: efficiency
  - Timing: scaling
- Can Use the CESM test suite to run MOBY tests
- Scientific Validation
  - To be determined (soon)

# Setting Up a New CESM MOBY Experiment

- Customize a version of CESM1
  - Check out copy of CESM1 from NCAR development repository
  - Swap out pop2 and scripts
- Set up a new MOBY case using standard CESM1 procedures
  - Presently, just the "ocean-only" version is supported, but implementing support for MOBY in the fully coupled version is straightforward
  - In the next slide, only the compset CDARWIN is nonstandard

# Setting Up a New CESM "Ocean-Only" with DARWIN Case

1. cd $CESM_MOBY/scripts
2. create_newcase -compset CDARWIN -res T62_gx3v7 -mach bluefire -skip_rundb -case $CASEDIR/$CASE
3. cd $CASEDIR/$CASE
4. configure -case
5. ./$CASE.build
6. bsub < $CASE.bluefire.run

# Next Steps

- Initial Conditions
- Final Round of Software Engineering Tests
- Final Walkthrough
- Scientific Validation