A New Flexible Coupler Designed for Earth System Modeling for CCSM4/CESM1

Tony Craig
Dec 16, 2010

NCAR Earth System Laboratory
CESM Software Engineering Group (CSEG)

NCAR is sponsored by the National Science Foundation
What is CCSM4/CESM1?

• Latest NCAR Climate Models being run in support of AR5
• CCSM = Community Climate System Model
• CESM = Community Earth System Model
  – CCSM4 + bgc, land ice, chemistry, waccm, new atm physics options, ...
• Coupled Atmosphere, Land, Ocean, and Sea Ice Components
• Diverse set of plug and play components (active, data, dead, stub)
• Runs at different resolutions and on various grids
• Exact restart
• Regular releases
Outline

- Role of couplers and features of coupled systems
- History of couplers at NCAR
- CCSM4 coupler implementation
- CCSM4 coupler performance
- Future challenges
- Summary
Role of “Couplers”

1) Communication of data between components
2) Sequencing and integration control
   • temporal sequencing
   • coupling frequencies
   • lags
3) Coupling methods
   • rearrange (redistribution) and mapping (regridding/interpolation)
   • merging of fields
   • computations (ie. flux calculation)
   • diagnostics
Features of Coupled Systems

- Single vs multiple executable
- Component layout on processors
  - sequential, concurrent, mixed
- “Hub” vs direct coupling between components
- Top level driver or coupling calls from inside components
Partial History of Climate Models at NCAR

- CCM0, CCM1, CCM2, CCM3 (1980-2000)
- CSM1 (1998)
- PCM (1998)
- CAM2 (2002)
- CAM3 (2004)
- CCSM4/CESM1/CPL7 (2010)
CCM / CAM

**CCM**
- 1980 - 2000
- ATM, LAND, prescribed OCN and ICE
- **All components on the same grid**
- **Single executable, sequential**
- No MPI, parallelization via CRAY multitasking, other methods, or not at all.
- **The ATM model was the driver and called surface models directly. The surface models were built by or for the ATM model.**

**CAM**
- 2000 – 2010
- ATM, LAND, prescribed OCN and ICE
- **MPI parallelization of all components**
- **All components on the same grid and same decomposition**
- Physical components consistent with CCSM2 and CCSM3. Coupling mechanism distinct from CCSM.
- Single executable, sequential, all components run on all processors
• 1998
• Prognostic ATM, LAND, OCN, ICE plus DRV/CPL
• Diverse grids
• Single executable, sequential, all components run on all processors
• MPI parallelization of all components
• Target platforms: CM5, T3D, T3E, SGI Origin
• Separate driver that called components without new interfaces, some entry points added. The components then “placed” their data in a “CPL” common block accessible to all.
CSM1 / CCSM2 / CCSM3

- **CSM1**
  - 1998
  - ATM, LAND, OCN, ICE and CPL
  - **Multiple executable, concurrent with separate coupler component**
  - Parallelization via shared memory on Cray Vector and SGI Origin
  - Cray Vector platforms supported processor sharing of executables (roll out)
  - **Coupler running on one processor**
  - Communication via PVM first then MPI
  - Separate hub coupler with **coupling calls made from inside each component**

- **CCSM2**
  - 2002
  - **Parallelization via MPI across active physical components**
  - **Coupler running on one processor**
  - Coupling via MPI
  - Target DSM (distributed shared memory) platforms – IBM SP, SGI

- **CCSM3**
  - 2004
  - **Parallelization via MPI across active physical components, some OpenMP**
  - Target DSM (distributed shared memory) platforms – IBM SP, SGI, Cray X1, Clusters, NEC
  - **Coupler parallelized with MPI using MCT infrastructure**
CCSM4 / CESM1

- 2010
- ATM, LAND, OCN, ICE, CPL, and DRV
- Single executable, flexible component processor layout capability
- Parallelization via MPI + OpenMP across all physical components
- Coupling via MPI
- Target platforms – IBM SP, Cray XT, Bluegene, Clusters
- Coupler parallelized with MPI using MCT infrastructure
- Separate hub coupler with coupling calls via standard interfaces through the driver
- DRV facilitating coupling between the coupler and other components
Motivation and Goals for CCSM4/CPL7 vs CCSM3/CPL6

- Better support for latest science including desire for tighter coupling
- Single processor, sequential capability
- Single executable
- Improved load balance/performance opportunity
- ESMF compatibility
- Higher resolution capability
- Improved memory and performance scaling
- Greater code reuse and reduced maintenance with respect to CAM “stand-alone” versions.
CCSM4 Coupler Implementation

- Single Executable
- Top level driver
  - Driver control of temporal sequencing
- Still hub and spoke in terms of the implementation
- Init/Run/Finalize interfaces in components
  - ESMF compatible option
- Flexible component layout on processors
- Improved memory and performance scaling
- Uses MCT, ESMF, and PIO
Interfaces

Consistent Init, Run, and Finalize interfaces across all components.

MCT Based Interfaces:
subroutine ocn_run_mct( EClock, cdata, x2o, o2x)
    type(ESMF_Clock) ,intent(in) :: EClock
    type(seq_cdata) ,intent(inout) :: cdata
    type(mct_aVect) ,intent(inout) :: x2o
    type(mct_aVect) ,intent(inout) :: o2x

ESMF Compatible Interfaces:
subroutine ocn_run_esmf(comp, import_state, export_state, EClock, rc)
    type(ESMF_GridComp) :: comp
    type(ESMF_State) :: import_state
    type(ESMF_State) :: export_state
    type(ESMF_Clock) :: EClock
    integer, intent(out) :: rc
Initialization

- Driver starts up on all processors
- MPI communicators are generated for each component based on namelist input. For each component (ie. ATM, LND, ICE, OCN, CPL)
  - number of MPI tasks
  - number of OpenMP threads per MPI task
  - root MPI task in global MPI space
- Driver calls the component initialization methods on specific processors and an MPI communicator is passed to each component
- Grid and decomp information from each component is returned to the driver
- The driver initializes rearrangers to communicate data between components and the coupler
- The coupler initializes various datatypes, rearrangers, mappers, etc.
- The driver begins the run loop

- The driver and coupler get all grid and coupling information from either namelist input or from the components directly at initialization.
Concurrency and Driver Sequencing

- The maximum amount of concurrency supported in CCSM4/CESM1 due to the driver implementation is

![Diagram showing concurrency and driver sequencing]

- The model results are identical* regardless of the processor layout of components because the temporal sequencing is set by the driver, not by the processor layout.

*with the caveat that changing processor counts for some components introduces round-off level differences
Component Layouts

All configurations running on 128 processors total

Concurrent: 33.5 s/day
Sequential: 20.8 s/day
Sequential plus ocean concurrent: 21.8 s/day
Mixed sequential concurrent: 19.1 s/day
Implementation Notes

- MCT is still used heavily in CPL7
- Mapping weights still generated offline using SCRIP
- Use of ESMF is being explored for weights generation as well as mapping performance and metadata.
- Use of the PIO (parallel IO) library developed in the CCSM community has been added to all components
  - Reduces memory use
  - Sits on top of netcdf3, netcdf4, pnetcdf, and MPI-IO
  - Always as fast or faster than “gather and write” or “read and scatter” methods using standard serial netcdf approach
  - Makes high resolution possible even with serial netcdf because of underlying parallel memory implementation.
Coupler Performance Kernels

1. Flux calculation
   - trivially parallel (no communication)
   - FLOPS (add,mult,divide,log,min/max,sqrt) dominates
2. Merge operation on ocean grid
   - trivially parallel
   - memory access dominates
3. Rearrange of ocean data between two decompositions
   - communication dominates
4. Mapping (interpolation) of data between atmosphere and ocean grids
   - combination of mult/add plus communication
Coupler Scaling Plots

- 2 resolutions
  - f19_g16 = 2 degree atm/Ind + 1 degree ocn/ice = 144x96 + 320x384
  - f05_t12 = 1/2 degree atm/Ind + 1/10 degree ocn/ice = 576x384 + 3600x2400
- 3 hardware platforms
  - bluefire = IBM SP6 at NCAR
  - jaguarpf = Cray XT5 at ORNL
  - intrepid = IBM BG/P at ANL
- log/log plots with same axes scales
- X axis is MPI tasks, range 1 to 10,000
- linear reference line
- time in seconds per simulated day
  - system run with “dead” components
  - 20 day runs, 48 calls/day, 960 calls/run
  - total time aggregated for each processor, barriers used
  - maximum time over all processors shown
  - on production machines
Scaling of Atm/Ocean Flux Calculation

2 degree atm/land + 1 degree ocean/ice

1/2 degree atm/land + 1/10 degree ocean/ice
Scaling of Ocean Merge Operation

2 degree atm/land + 1 deg ocn/ice

1/2 degree atm/land + 1/10 deg ocn/ice
Scaling of Ocean Rearrange

2 degree atm/lnd + 1 deg ocn/ice

1/2 degree atm/lnd + 1/10 deg ocn/ice
Scaling of Atm to Ocean Mapping

2 degree atm/Ind + 1 deg ocn/ice

1/2 degree atm/Ind + 1/10 deg ocn/ice
Coupler Performance Summary

- Flux calculation scales well on all machines.
- Merge operation doesn’t scale as well especially as the number of gridcells per processors becomes small.
- Rearrange tends to scale well at lower processor counts but poorly at higher processor counts with some hardware exhibiting roll-over.
- Mapping operation scales similar to the rearrange operation suggesting map scaling is dominated by communication cost.
Future Challenges to Increasing Model Throughput
(ie. Reducing Time to Completion)

• In the past decades, we have improved throughput via
  – getting faster hardware on a per processor basis
  – improving grid decomposition strategies for use with MPI
• Challenges in the future because
  – Processors are not getting much faster anymore
  – Memory per processor is not increasing much anymore
  – Processors per node and total processors available are increasing rapidly
  – Future hardware may be highly non-homogenous (GPUs, etc)
  – Scaling has never been one of the strongest features of climate models
  – Higher resolutions and more complex science always results in slower models
• Solutions?
  – Faster hardware networks?
  – More levels or parallelism in models?
    • Component (ATM, OCN, etc), Gridcells, Tracers or Physics, ?
  – New parallelism paradigms?
    • One-sided communication
    • Overlap work and communication better
    • New communication implementations?
Summary

- New CPL7 implementation for CCSM4/CESM1
- Improves overall model throughput/cost via more flexible component processor layout capability
- Memory and performance scaling are better than previous implementations
- Demonstrated ability to run coupler on 10k processors.
- Work still to be done in the coupler
  - Scaling and tuning of communication intensive routines
  - IO performance tuning
- Future performance challenges
- CPL7 supports high resolution capability
  - Demonstrated 1/4+1/10 and 1/8+1/4+1/10 configurations on 10k to 100k processors
- Next set of features driven by science interests
The End
Driver Sequencing

Do (run the model)
   Call CPL_to_OCN
   Call CPL_to_LND
   Call CPL_to_ICE
   Call OCN_RUN
   Call LND_RUN
   Call ICE_RUN
   Call ATM_OCN_FLUX
   Call ICE_to_CPL
   Call LND_to_CPL
   Call CPL_to_ATM
   Call ATM_RUN
   Call ATM_to_CPL
   Call OCN_to_CPL
Enddo
CAM2, CAM3

- 1998 – 2010
- ATM, LAND, prescribed OCN and ICE
- Physical components consistent with CCSM2, and CCSM3. Coupling mechanism distinct from CCSM.
- Single executable, sequential, all components run on all processors
- MPI parallelization of all components
CSM1

- 1998
- ATM, LAND, OCN, ICE and CPL
- Multiple executable, concurrent with separate coupler component
- Parallelization via shared memory on Cray Vector and SGI Origin
- Cray Vector platforms supported processor sharing of executables (roll out)
- Coupler running on one processor
- Communication via PVM first then MPI
- Separate hub coupler with coupling calls made from inside each component
CCSM2/CPL5

• 2002
• ATM, LAND, OCN, ICE, and CPL
• Multiple executable, concurrent with separate coupler component
• Parallelization via MPI across active physical components
• Coupling via MPI
• Target DSM (distributed shared memory) platforms – IBM SP, SGI
• Coupler running on one processor
• Separate hub coupler with coupling calls made from inside each component
## Driver Sequencing
(Update This!!)

<table>
<thead>
<tr>
<th>Driver Operation</th>
<th>CPL</th>
<th>LND</th>
<th>ICE</th>
<th>ATM</th>
<th>OCN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ocean Prep</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPL to OCEAN communication</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Land Prep</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPL to LAND communication</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sea Ice Prep</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPL to ICE communication</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ocean Run</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Ice Run</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Land Run</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Atm/Ocean Surface Flux Computation</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICE to CPL communication</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ice Post</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAND to CPL communication</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Land Post</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atm Prep</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPL to ATM communication</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atm Run</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>ATM to CPL communication</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atm Post</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCEAN to CPL communication</td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ocean Post</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>