



Argonne
NATIONAL
LABORATORY

... for a brighter future



U.S. Department
of Energy

UChicago ►
Argonne_{LLC}



**Office of
Science**
U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC

On the Road to a Sequential CCSM

Robert Jacob, Argonne National Laboratory

*Including work by: Mariana Vertenstein (NCAR), Ray
Loy (ANL), Tony Craig (NCAR)*

SEWG Meeting, March 16th, 2007



Concurrent

Sequential



Taxonomy of Model Integration Schemes

- Execution (a computer instruction view)
 - concurrent: completely different execution sequences are occurring on different processors. Inherently parallel.
 - Sequential: same instruction sequence on all processors

- Integration (a science view)
 - Sequential (in phase): each model integrates over the same time period before advancing to the next time period
 - Staggered (out of phase): one or more models are lagged.

CCSM models in this taxonomy

- Concurrent execution, staggered integration:
 - current CCSM (single or multiple executable)
- Sequential execution, sequential integration:
 - current CAM
- Sequential execution, staggered integration:
 - **sequential CCSM** and CAM-mode of sequential CCSM.
- Concurrent execution, sequential integration:
 - Possible, but has pathologically unbalanced load.

Motivation for Sequential CCSM

- 2-64 processors
 - Only option for 2-4
 - Tough to load balance concurrent on small processor counts
 - Trivial to load balance sequential execution system.
 - 2-64 processors coming soon to your next laptop/workstation!
- High resolution/1000's of processors
 - Again, trivial to load balance assuming scalability
 - Impossible to remove all idle time from concurrent case; this is costly when 1000's of processors are idle.
 - Platforms like BlueGene will be used for high-resolution runs and ensembles (100 64-proc runs).

Sequential CCSM: Current Status

- seq_ccsm_drv.F90
 - Main driver for sequential CCSM
 - MCT based. ESMF-based driver in development
 - Declares all states (AttributeVectors), grids (GeneralGrids), and decomposition descriptors (GlobalSegMaps)
 - Declares and initializes all mappings.

Sequential CCSM: Current Status

- seq_ccsm_drv.F90
 - For models, driver calls.
 - *<model>_init_mct (initializes states, decomp, grids)*
 - *<model>_run_mct (read/write states)*
 - *<model>_final_mct (clean up)*
 - *Model developers must implement these calls to be coupled with sequential driver.*
 - Driver also calls all mappings, fluxes, merges.

Current CAM trunk uses seq_ccsm_drv to call CAM, CLM, DOM/SOM, CICE.

Works with a single resolution.

Sequential CCSM: Development with dead code

■ dead7

- New code that provides functionality of CCSM's dead models for sequential CCSM.
- Can mimic any resolution and decomposition
- Receives/sends same states as full model
- Dead7 is being used to test different resolutions in sequential CCSM.

Sequential CCSM: multiple resolutions

- Same map routine supports same or different resolution for cases where different resolution is allowed: atmocn, atmice, rofocn
- Simple test for same resolution based on global grid size:

```
icesize=mct_gsMap_gsize(gsMap_i)  
atmsize=mct_gsMap_gsize(gsMap_a)  
samegrid=.false.  
if(icesize .eq. atmsize) samegrid =.true.
```

Sequential CCSM: multiple resolutions

- If different grids, a *config* or *resource* file is read during mapping init to determine mapping weight filename and mapping type:

(excerpt from seq_maps.rc)

```
atm2ocnFmapname: map_T31_to_gx3v5_aave_da_040122.nc  
atm2ocnFmaptype: X  
  
atm2ocnSmapname: map_T31_to_gx3v5_bilin_da_040122.nc  
atm2ocnSmatype: X
```

- Currently working with dead models.

Sequential CCSM: other recent advances

- datm7 (single-node version) added to driver
- CICE also integrated.
- dead7 land with river model introduced and river mapping called from driver

CCSM and memory scaling

- Both **sequential** and **concurrent** execution CCSM may be run on BlueGene
- BlueGene, and similar planned machines, have low per-node memory.
- What is the memory scaling of the coupler?
- Early suspect: reading in mapping weights
 - Concurrent and sequential use same mapping scheme.

Mapping memory issues

■ Old Mapping Algorithm

- Allocate memory for all non-zero interpolation weights (ns) and all area weights for each grid (na, nb)
- Read in all data to node 0
- Scatter all data to other nodes
- River to gx3: 242 MB!

■ First New Algorithm (Rob)

- Allocate weights and areas one at a time.
- Read and scatter one at a time
- Deallocate between each set.
- River to gx3: 190MB.

Mapping memory issues

- Second New Algorithm (Tony Craig)
 - Allocate a user-adjustable amount of memory
 - Read in an equivalent amount of weights
 - Broadcast to all nodes
 - Each node finds its piece
 - Repeat until all weights read.
 - Tests show this to be faster than read-all-and-scatter

Investigate coupler memory scaling on BlueGene

Use concurrent CCSM with dead models (single executable)

- In all cases, dead models given 1 processor each, others all given to coupler
- “32” = 28 coupler processors
- “512” = 508 coupler processors
- T42_gx1: runs to completion: 32,64,...,512
 - 1024:
 - *ran out of memory in mapping init with old algorithm*
 - *New algorithm: gets past map init. Some make it to main loop. Others die in frac_set.*

Investigate coupler memory scaling on BlueGene

Use concurrent CCSM with dead models (single executable)

- T85_gx1: similar results

- T340_x01:
 - Dies in initialization before reaching map init
 - Problems with trapping exit on BlueGene.
 - Dead models appear to be running out of memory even though diagnostics indicate plenty of free memory.

Another suspect: extra memory for different mappings

■ In cpl6:

- Coupler receives all data from atmosphere
- States and Fluxes are each copied into new datatypes and follow different paths through coupler
 - *States are mapped with bilinear mapping, fluxes with conservative mapping.*
 - call `cpl_map_bun(atm_states_a, bilinear_map, atm_states_o)`
 - call `cpl_map_bun(atm_fluxes_a, conserv_map, atm_fluxes_o)`
- Results in 2x memory hit compared to just keeping one copy of received data.

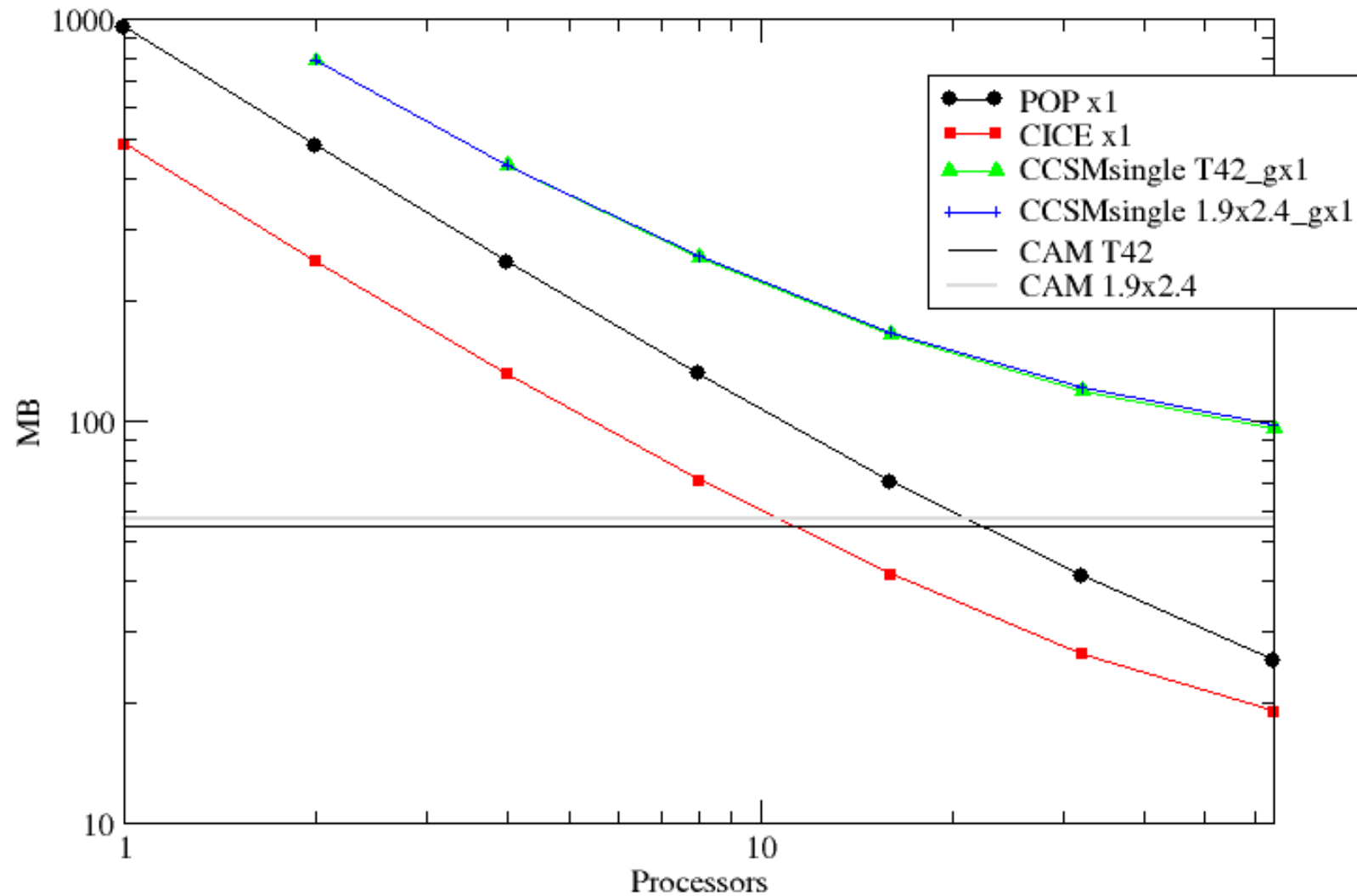
■ In sequential CCSM:

- Use new MCT feature which allows attributes to be specified
 - call `map_atmocn_mct(atmfields_a, bilinear_map, atmfields_o, fields_to_mapb)`
 - Call `map_atmocn_mct(atmfields_a, conserv_map, atmfields_o, fields_to_mapc)`

Live models on BlueGene (ccsm 3.1.beta41)

- T31_gx3 runs without modification! On 32 BlueGene processors (smallest amount available; 12 atm, 12 ocn, 4 ice, 2 Ind, 2 cpl) and up to 256 (not load balanced).
- Note: BlueGene has no dynamic libraries. Everything must be statically linked.
- First attempt at 1.9x2.4_gx1: 1.5GB executable!
 - Was using same decomposition as dead model studies: 1 processor for POP, CICE.

CCSM executable image size on BlueGene



CAM on BlueGene

- T42: runs fine up to 64 proc limit. CO and VN mode
- T85: runs fine up to 128 procs in CO mode, VN has memory problems
- FV 1x1.25: CO runs on 32,64,128,256, and 480, FV has memory problems
- Higher resolutions: needs parallel I/O

CCSM sequential: next steps

- Add time averaging
- Use same low-memory map read as cpl6
- Add other cpl6 functionality: area normalization, diagnostics.
- Check scaling of executable image size on BlueGene
- Integrate other data models
- Integrate POP2