Experience with the Full CCSM

Matthew Cordery and Ilene Carpenter (Cray),
John Drake and Pat Worley (ORNL)
Outline

• CCSM overview
• Porting strategy for coupled model
• Porting issues
• CAM/CLM optimization and performance
• Configuration
• Performance
CCSM overview

• CCSM, the Community Climate System Model is a coupled model for simulating the earth’s climate system.
  – Developed at NCAR with significant collaborations with US DoE, NASA and the university community

• Components include
  – Atmospheric Model – CAM 2.0.2
  – Ocean Model – POP 1.4.3
  – Sea Ice Model – CSIM4
  – Land Model – CLM2
  – coupler
CCSM Components

CAM

CLM

coupler

CSIM

POP
Porting Strategy

• Individual components vectorized by a number of organizations including NCAR, ORNL, ARSC, Cray, NEC and Earth Simulator

• Simultaneously, port coupled system framework, which includes coupler (cpl6) and utilities it uses:
  – MCT – Model Coupling Toolkit from ANL
  – MPEU – Message Passing Environment Utilities from NASA DAO
  – MPH – Multi Program Handshaking Utility from LBL
Porting Issues

- CAM needs to be compiled with \(-s\) real64 to run correctly
- This means libraries and all component models need to be built with \(-s\) real64
- Word length issues (double precision) in utilities
- Minor MPI Word length issue in POP, which in standalone code is *not* compiled with \(-s\) real64
Porting Strategy

- Build with new multiple binary capability in Cray MPI library.
- Use “data” models to exercise coupling framework without real models
  - read data from files and communicate with coupler.
  - datm, dlnd, docn, dice, cpl
- Add real models one at a time to debug
  - CAM, dlnd, docn, dice, cpl
  - CAM, CLM, docn, dice, cpl
  - datm, dlnd, POP, dice, cpl
  ...
Performance Optimization

- Land and Ice models already optimized by other groups.
- Standalone POP has been optimized for X1 but don’t expect to need to use large number of processors on POP in CCSM so may not need to include those mods.
- Expect performance of CCSM to be determined primarily by performance of the atmospheric model (CAM) so focus attention on it. Target of 20-25 simulated years per day for T85 atmosphere.
CAM Porting Issues

• Mostly system calls and macro definitions
  - E.g. `getenv()` ⇒ `pxfgetenv()`
  - Define UNICOSMP macro
Optimization Rules and Issues

- Cannot impact performance on other target systems
- Solution must be independent of # procs
- Cannot alter solution (bit-for-bit) on other platforms
- Limited amounts of architecture-dependent code allowed (i.e. no large scale #ifdef NEC/CRAY/IBM sections)
- Frequent updates to models
CAM Optimization Hotspots

- Physics
- Dynamics
- Land Model
- Communications

As with many environmental applications, initial profiles were relatively flat.
Physics Optimizations: Hotspots

• Hotspots (easy-to-hard)
  – Function calls within loops
    • estblf() – saturation pressure lookup
  – Error checks with I/O
  – Short/long-wavelength radiation routines
    • Not streamed/vectorized
    • Complex cloud overlap algorithm
    • Few opportunities for long vectors
Physics Optimizations: Inlining

• Function calls within loops
  – Estblf is called very often and its presence in loops inhibits vectorization and streaming.
  – Fixed with –Omodinline in certain modules
    • Default behavior in newest compilers
Physics Optimizations: I/O

• Error checks with I/O
  
  ```plaintext
do i = 1, N
    err = f(i) - g(i)
    if( err > tol ) then
      write(6,fmt) msg, i, err
      call endrun()
    end if
  end do
  ```

• Presence of write statement forces loop to be scalar.

• Call to endrun() inhibits streaming.
Physics Optimizations: I/O

• Not the same, but it streams/vectorizes....

\[
j = 0; \ jerr = 0.0
\]
\[
do \ i = 1, N
\]
\[
\quad \text{err} = f(i) - g(i)
\]
\[
\quad \text{if}( \ \text{err} > \text{tol} ) \ \text{then}
\]
\[
\quad \quad j = i; \ jerr = \text{err}
\]
\[
\quad \text{end if}
\]
\[
\text{end do}
\]
\[
\text{if}( \ j > 0 ) \text{then}
\]
\[
\quad \text{write}(6,*) \ \text{msg, err, j}
\]
\[
\quad \text{call endrun()}
\]
\[
\text{endif}
\]

• Done in qneg3, aerosols, etc.
Physics Optimizations: radclwmx

• Complex cloud algorithm limits vectorization
• $DIR CONCURRENT for loops with indirect addressing, e.g. i = indx(j)
• Forced streaming over number of columns.
  – Amount of work still less than optimized short wavelength code.
  – Streamed within radclwmx rather than at a higher level
Physics Optimizations: radcswmx

• **First pass:**
  – Vectorized across spectral bands
  – Forced streaming across number of columns
  – Very simple to implement and gives good performance boost on X1.

• **Problem:**
  – Short vector lengths (19) means relatively inefficient performance compared to vectorizing over daylight columns.
  – Inefficient implementation for machines that need long vectors.
• Second pass:
  – Developed by NEC
  – Introduce new data structures and routines that assist in vectorizing over the number of daylight columns.

• Problem:
  – Additional complexity. Compress-expand overhead.
  – No significant performance boost on X1 over previous version.
  – Still some bottleneck loops with short vector lengths.
Physics Optimizations: load balancing

• Turned on load balancing option already in code.
  – Unlike other platforms, this pays off on X1
Dycore Optimizations

- While the physics scales well to high processor counts, the spectral dycore did not.
- A number of issues needed to be addressed:
  - Sub-optimal packing/unpacking before communications
  - Serial communications
    - Use all-to-all or allgather
  - Load imbalance caused by streaming of work-critical loops with loop lengths less than four.
    - Move streaming to loops with more work, e.g. loops over number of latitude bands
Communications Optimizations

- Co-Array Fortran versions of MPI wrapper routines
  - Streamed and vectorized
  - Used pointer structure keeps memory requirements the same and allows use of co-arrays.
  - Additional barriers but offset by faster point-to-point communications.
  - Need to determine whether benefit outweighs goal of minimizing platform-specific code.

- MPI optimization
  - More all-to-all communications, less one-to-all and all-to-one communications.
CAM T42 (dev50) Performance

- Cray X1

Graph showing simulated years per wall day against processors.
CAM T85 (dev50) Performance

![Graph showing performance over processors]

Simulated Years/Wall Day

Processors

Cray X1
CAM performance and versions

- Most of the optimization modifications in CAM/CLM are in the latest CCSM3 source.
- CAM dev70 runs about as fast as dev50.
Land Model

• Original CLM2.2 contained data structures that were inherently ‘vector unfriendly’
  – The internal data structures were based on a hierarchy of pointers to derived data types containing scalar quantities scattered throughout memory.
  – Lowest level loops over ‘plant functional types’ with max loop lengths of 1-20 and snow/soil loops with negligible work.
Land model optimization

- Develop a single code that runs well on both vector and scalar architectures while maintaining the hierarchical nature of the current data structures.
- Move loops over columns into the science subroutines, and vectorize over these outer loops (instead of the short inner loops over PFTs and soil/snow levels).
- Unroll short loops, interchange some loops, fuse some loops, and inline subroutines to improve performance.
New land model performance

• smaller memory footprint
• new data structures simplify history updates and reduce complexity and # of gather/scatters
• 25.8x faster on the Cray X1, and 1.8x faster on the IBM
Coupler

- Small number of porting mods needed in utilities used by coupler to deal with word length and auto-promotion.
- No X1 specific optimization done.
Configuration Optimization/Plan

• Optimal performance of CCSM requires determining how to distribute processors among 5 executables
• Expect to run CAM with 128 processors to maximize number of simulated years per wall day.
• Expect to use smaller numbers of processors on other components (8, 16, 24) – just enough to not slow down the atmospheric model
• Initial runs have been made but final configuration (number of processors for each component) has not yet been determined. T85 runs used
  – CAM 128, 64 or 32 MSPs
  – POP 24 MSPs
  – CLM 12 MSPs
  – CSIM4 8 MSPs
  – Cpl6 8 MSPs

• Initial performance is about 6-7x slower than expected.
  – Coupled model performance should be close to standalone CAM performance.
  – Have not yet analyzed results to determine bottleneck.
  – Ran with timers on, no modinline (because of build issue with coupled system) and with some streaming disabled in land model.
Future Plans

- Validation of climate (NCAR).
- Identification and elimination of performance problems that affect fully coupled runs.
  - Examine overhead of coupler, determine if additional optimization is needed.
  - Examine performance of POP in coupled system, determine which mods from optimized standalone code may be needed in coupled model.
- Load balancing of coupled system.
Summary

• The full CCSM has been ported to the X1.
  – Makefiles, scripts and many source code mods will be in next release.

• Significant optimization of each component has been done by groups at Cray, ORNL, NCAR and NEC.

• Performance of individual components is excellent.

• Initial performance of coupled model is currently poor.
  – Coupled model with vectorized components has only been available for a few days.
  – Some compiler optimizations were turned off because of issues building coupled system.
  – Expect this to be fixed within a few weeks.