Adventures in Vectorizing the Community Land Model

CUG 2004

Forrest Hoffman, ORNL
Mariana Vertenstein, NCAR
James B. White III (Trey), ORNL

Acknowledgement

Research sponsored by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

CLM data model

- Grid cells
  - Same horizontal grid as CAM
- Land units
  - Percentages of grid cell, not contiguous region
  - Soil properties, landcover types (lakes, glaciers)
- Columns
  - Soil states, fluxes with atmosphere
- Plant functional types (PFTs)
  - Compete for column resources

CLM vectorization history

- CLM 2.1 introduced “vector-hostile” data structures
- Experiments with simple vector data structure presented at CUG 2003
- CLM re-implemented with compromise data structure over last year
- Ongoing performance optimization

CLM 2.1 challenges

- High-level loops over columns
- Deep call stacks passing single-column arguments
- Very short inner loops
  - PFTs (currently length 1, 1-20 in future)
  - Soil and snow levels (single digits negligible work)
- Data structure…

CLM 2.1 data structures

- Nested user-defined types
  - Embedded pointers to sub-types
  - Arrays of pointers to sub-levels
  - Pointers up to parent level
  - Scalar quantities
- Subroutine structure
  - Pass single column argument, "c"
  - Declare (many!) local pointers
  - Associate local pointers with contents of "c"
  - Compute with local pointers as shorter aliases

CUG 2003 CLM experiment

I did this! Other authors did everything else.

- Implemented new data structures for one major physical process
- No user-defined types
- Major types become modules
  - Grid cells, land units, columns, PFTs
- "Flatten" remaining types
  - Scalar variables become 1D arrays
  - Arrays add a leading dimension
- Illustrated need for filters (index arrays) to replace if statements

CLM3.0 data structures

- Arrays of derived types → derived types with (pointers to) arrays
- Retained type hierarchy
  - One instance of each type
- Clumps and filters
- Required major rewrite of CLM

Clumps for data decomposition

- Independent grid cells (coupled through atmosphere)
- Grid cells distributed cyclically into “clumps”
- Clumps distributed cyclically among processes
- Clumps sized for:
  - Cache blocking
  - Vector length
  - Load balancing
  - Shared-memory parallelism (OpenMP and/or CSD)

Filters for vectorization

- Implemented as index arrays
- Group columns and PFTs based on relevant physical processes
  - Snow/non-snow, lake/non-lake, bare soil
  - PFTs that use dynamic vegetation
- Some filters created once
- Dynamic filters for snow/non-snow and vegetation

CLM3.0 driver

- Each MPI process loops over clumps
  - OpenMP or CSD parallelism
  - Using both would require two loops
- Arguments passed to physics routines:
  - Clump bounds for relevant levels of type hierarchy (grid cell, land unit, column, PFT)
  - Relevant filters
- Inner loops over clump elements
  - Sometimes filtered
  - Clump arrays implemented with pointers requiring "`!dir$ concurrent`"

Vectorization process

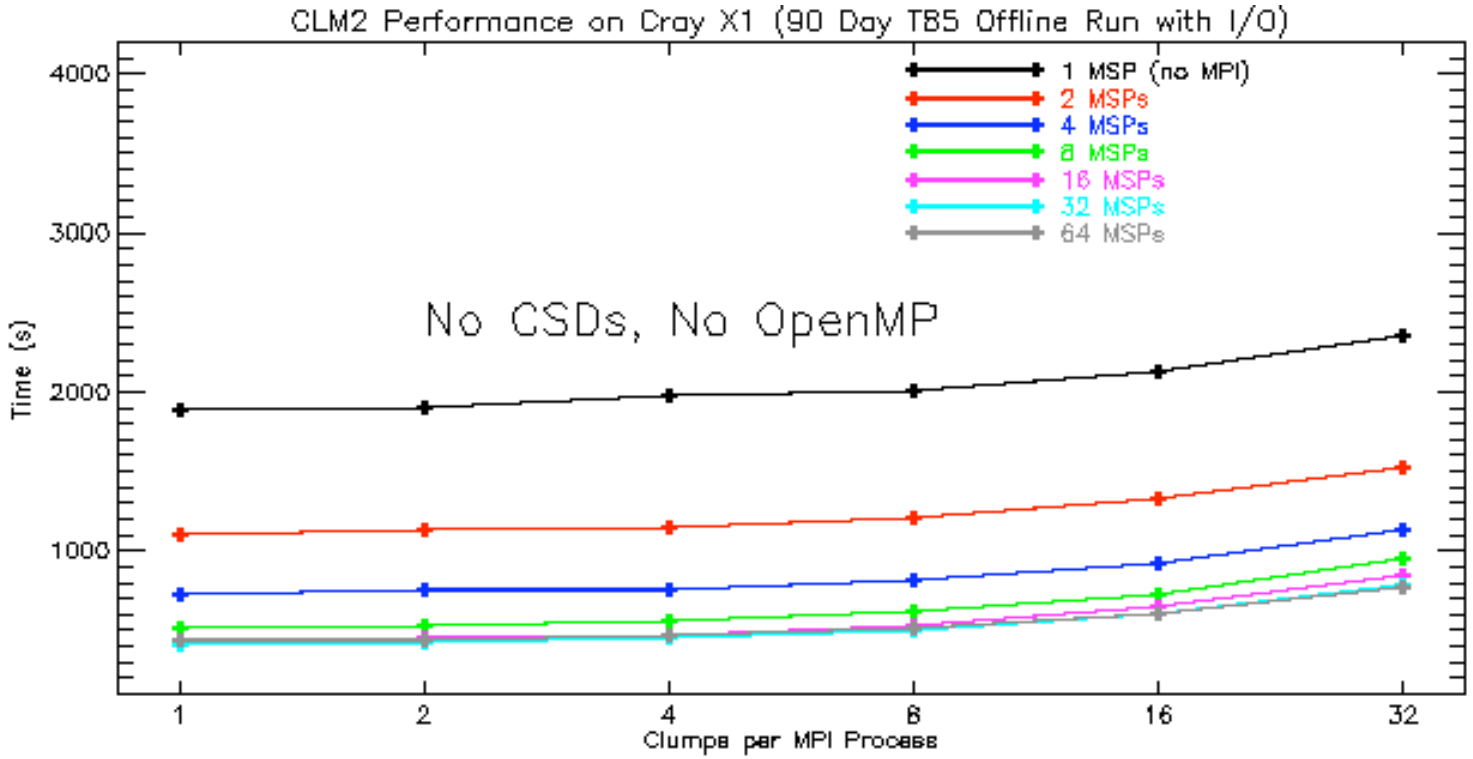
- New data structures and vectorization strategy approved by Land Model Working Group and NCAR researchers
- Created vector development branch in NCAR CVS repository
- Modified one subroutine at a time
  - Tested for correctness and performance on X1 and IBM Power4
- Cooperated with CRIEPI (Earth Simulator)
  - Provided protoype code
  - Received compiler listings and suggestion from Dave Parks of NEC

Vectorization results

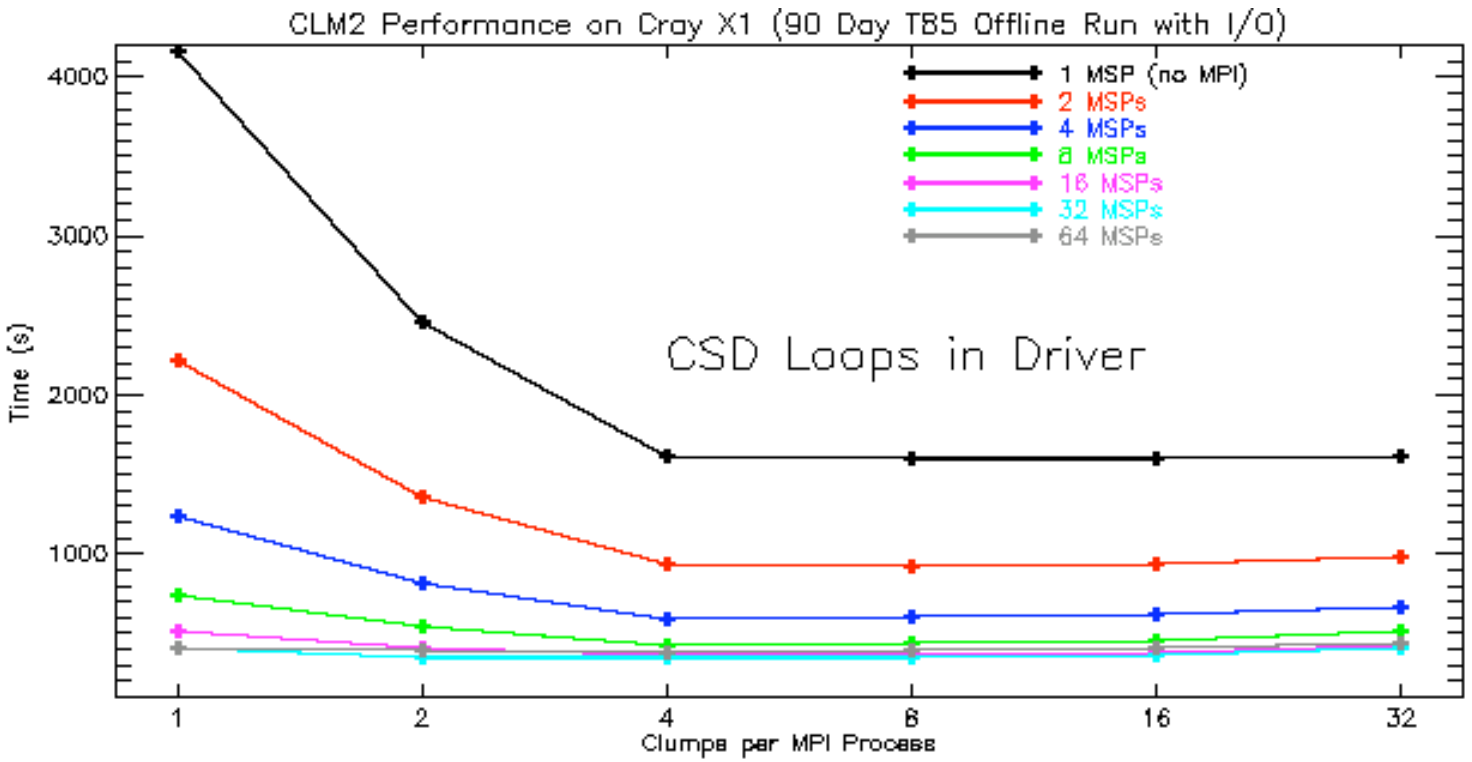
- Completed in October 2003
- CLM3.0 faster than CLM2.1
  - Improved 25.8x on X1
  - Improved 1.8x on IBM Power4
  - Smaller memory footprint
  - Simpler history updates
  - Simpler and fewer scatter/gathers

Performance comparisons

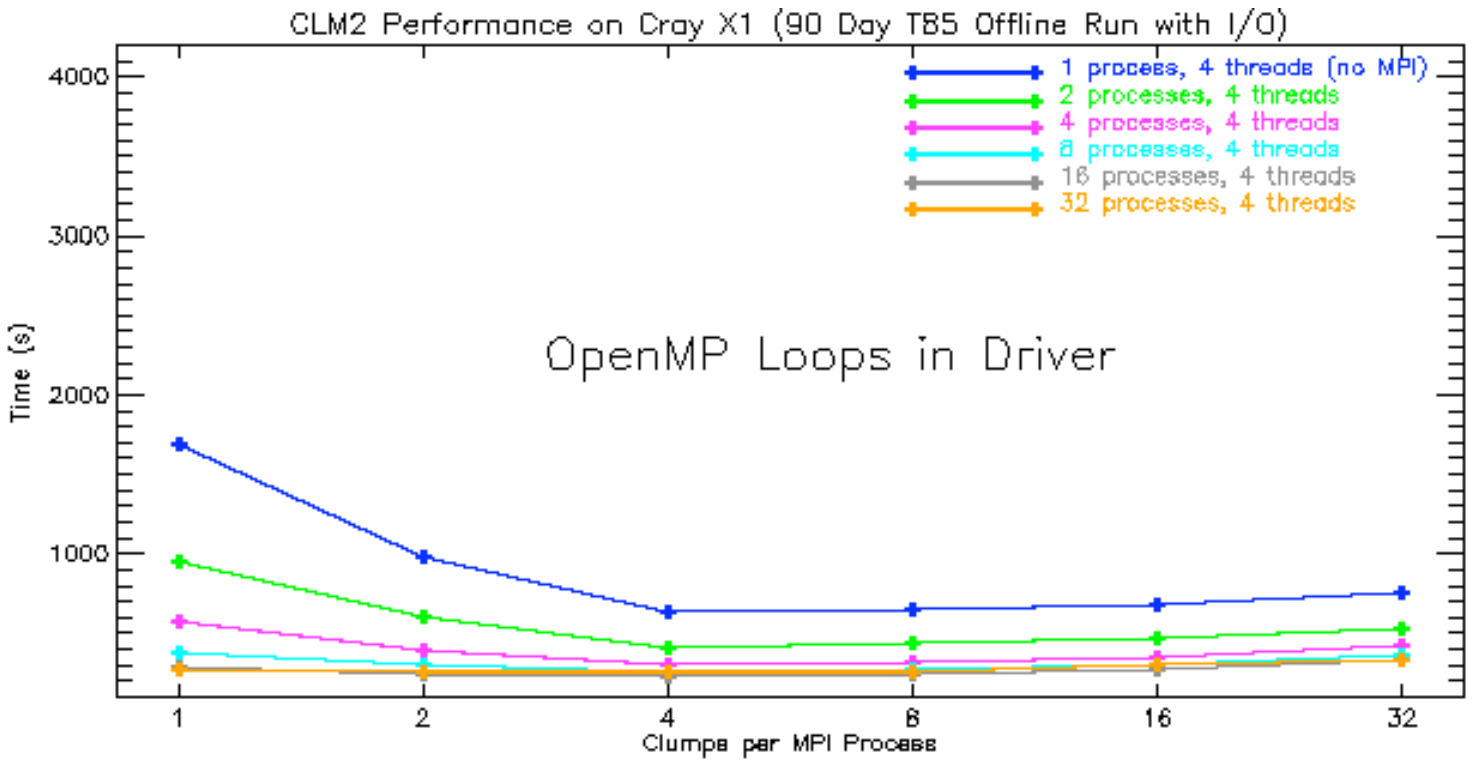
- Stand-alone CLM3.0
  (though plots say CLM2)
- 90 simulated days at T85 resolution
- 1,2,4,8,16,32,64 MSPs
- 1,2,4,8,16,32 clumps per MPI process
- Different shared-memory parallelism
  - Automatic multistreaming only
  - CSDs in driver
  - OpenMP in driver (with automatic multistreaming)



- Scales to 32 MSPs
- One clump per process works best

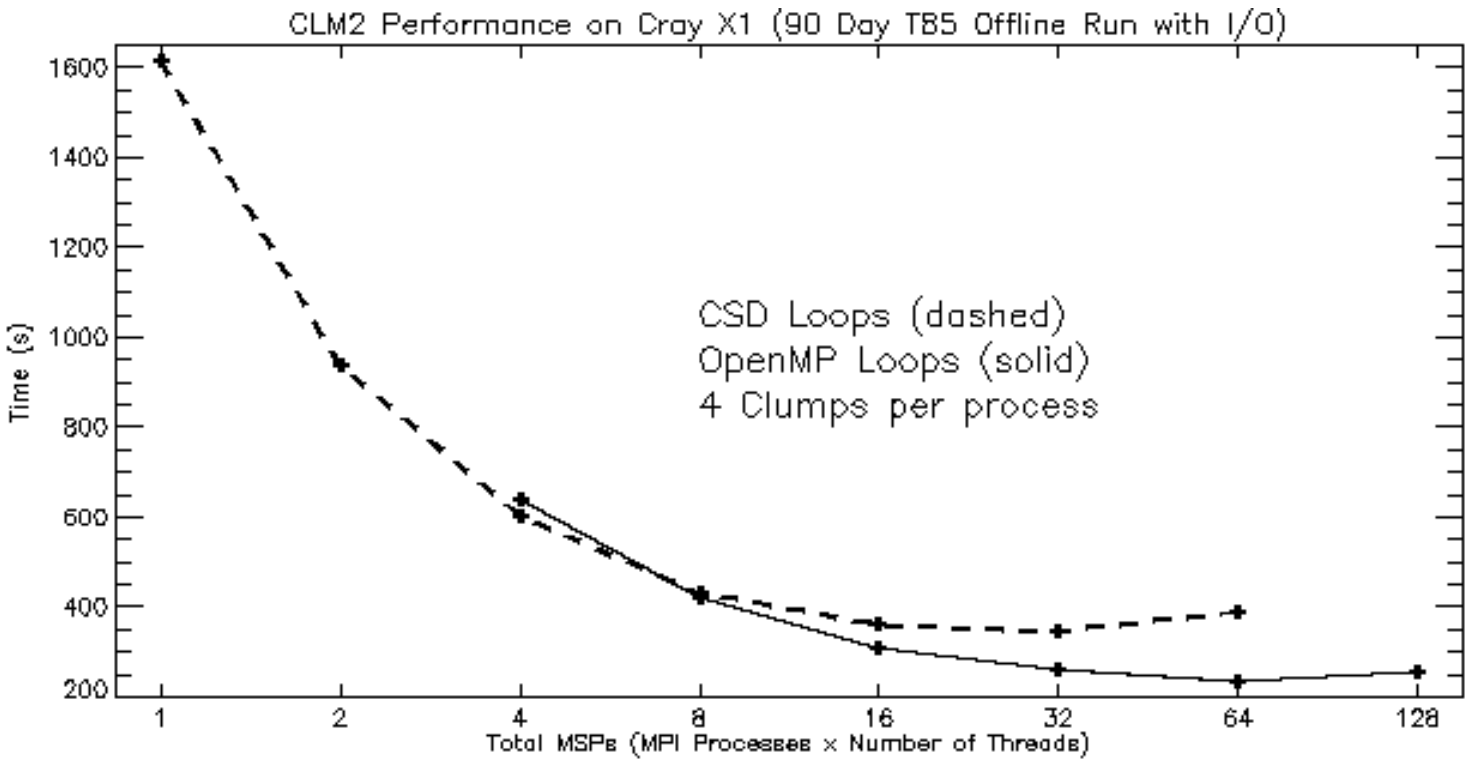


- Again, scales to 32 MSPs
- Need at least four clumps per process
  - One for each SSP



- Scales to 64 MSPs (16 MPI processes)
- Four MSPs per MPI process
  - Need at least four clumps

OpenMP vs. CSD



OpenMP vs. CSD, rematch

- CSDs removed January 6
- Restored April 27 because CCSM won't use OpenMP on X1
  - Multiple-binary MPI on X1 requires every process uses same number of threads
  - POP not threaded
- OpenMP & CSD?

Trey's conclusions

- Arrays of type hierarchies are bad for performance
  - Inhibit dependence analysis (and thus vectorization)
  - Reduce spatial locality
- Types with pointers to arrays require directives
  - Less safety, may camouflage real dependencies
  - Cray Fortran compiler allows allocatables (but others don't yet)
- Lessons for language design
  - Changing a data structure can cause profound code churn
  - The "filter" concept needs integrated support