# Vectorizing the Community Land Model Version 3.0 (CLM3.0)

*Forrest M. Hoffman, Mariana Vertenstein[*], Hideyuki Kitabata[+], J. B. White III, Patrick Worley, John Drake, Matthew Cordery[x]*

*Oak Ridge National Laboratory, [*]National Center for Atmospheric Research, [+]Central Research Institute of Electric Power Industry (Japan), and [x]Cray Inc.*

## The Vectorization Challenge

The Earth Simulator in Japan and the Cray X1 at Oak Ridge National Laboratory have sparked renewed interest in vector computers among researchers using the Community Climate System Model (CCSM).


To utilize these systems, all the CCSM component models required software engineering efforts to vectorize each code.

The Community Land Model (CLM) was particularly vector hostile and required a complete re−write to provide acceptable performance on vector architectures.


The strategy for CLM vectorization was:

- Develop a single CLM code that runs well on both vector and scalar machines while maintaining the hierarchical nature of the existing data structures.
- Alter data structures to obtain short and predictable data strides.
- Move loops over columns into science subroutnes, and vectorize over these long loops instead of the short loops over nested PFTs and soil/snow levels.
- Create readable and understandable code while avoiding vector−specific versions of code (#ifdef's) everywhere possible.
- First, do no harm! Code changes must not reduce the performance on the present target scalar platforms.

## New Data Structures in CLM3.0

- In CLM, the horizontal land surface heterogeneity is represented by a nested subgrid hierarchy of gridcells, landunits, columns, and plant functional types (PFTs).
- The hierarchical subgrid organization is reflected in the data structures used in the model code.
- The grid hierarchy was previously implemented as arrays of derived data types containing scalars for flux and state variables.


**Gridcells:** Computational grid shared with the atmosphere model.

**Landunits:** Geomorphologically distinct land cover types (glacier, lake, crop, urban, etc.).

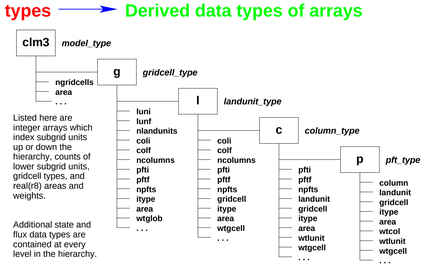
**Columns:** Water, snow, and soil state variables.

**Plant functional types:** Vegetation state variables. PFTs may compete for column−level resources.

- The new data structures represent the grid hierarchy as derived data types containing arrays for flux and state variables. In Fortran 90, these arrays are implemented as pointers.

Arrays of derived data types → Derived data types of arrays

- A cache−friendly blocking structure, called clumps, is superimposed on the data structures for improved computational efficiency.
- For optimal load balancing, gridcells are distributed among clumps in cyclic fashion, and clumps are distributed among MPI processes in cyclic fashion.
- Clumps also serve to block data for shared memory parallelism using OpenMP or Cray streaming.




- Another set of super−structures, called filters, were added to better support vectorized processing of columns and PFTs.
- Filters group like columns or PFTs based on their process−specific categorization and are used for indirect addressing into the main data structure hierarchy.

## Validated on the Cray X1

- ✔ Offline Community Land Model (CLM3.0)
- ✔ Offline CLM3.0 with Dynamic Vegetation Model (DGVM)
- ✔ Standalone Community Atmosphere Model (CAM3.0) with CLM3.0
- ✔ Standalone CAM3.0/CLM3.0 with DGVM

## PHOENIX


## Cray X1

- 128 SMP nodes
- 4 Multi−Streaming Processors (MSPs) per node
- 4 Single Streaming Processors (SSPs) per MSP
- Two 32−stage, 64−bit wide vector units running at 800 MHz and one 2−way superscalar unit running at 400 MHz per SSP
- 2 MB E−cache per MSP
- 16 GB of memory per node

512 processors (MSPs), 2048 GB of memory, and 6400 GFlop/s peak

## Code Reorganization

- The highest level loops in the main driver routine run over clumps for each MPI process and provide for OpenMP or Cray Streaming parallelism.
- Science subroutines called within these loops are passed local clump bounds for gridcells, landunits, columns, and PFTs as needed.
- Also passed are relevant filters (counts and vector indices).

```
      nclumps = get_proc_clumps()
!$OMP PARALLEL DO PRIVATE (nc,begg,endg,begl,endl,begc,endc,begp,endp)
!CSD$ PARALLEL DO PRIVATE (nc,begg,endg,begl,endl,begc,endc,begp,endp)
      do nc = 1,nclumps
         call get_clump_bounds(nc, begg, endg, begl, endl, begc, endc, begp, endp)
           .
           .
         call Hydrology1(begc, endc, begp, endp, &
              filter(nc)%num_nolakec, filter(nc)%nolakec, &
              filter(nc)%num_nolakep, filter(nc)%nolakep)
           .
           .
      end do
!CSD$ END PARALLEL DO
!$OMP END PARALLEL DO
```

- Loops within science subroutines run over grid or subgrid units.
- Many loops over subgrid units use filters for indirect addressing.
- The use of pointers in data structures requires compiler directives for loop vectorization.

```
      ! Assign local pointers to derived type
      ! members (landunit−level)
      clandunit => clm3%g%l%c%landunit
      itype     => clm3%g%l%itype
      ! Assign local pointers to derived type
      ! members (column−level)
      cgridcell => clm3%g%l%c%gridcell
      t_grnd    => clm3%g%l%c%ces%t_grnd
      h2osno    => clm3%g%l%c%cws%h2osno
      snowdp    => clm3%g%l%c%cps%snowdp
      snowage   => clm3%g%l%c%cps%snowage

   !dir$ concurrent
   !cdir nodep
      do f = 1, num_nolakec
         c = filter_nolakec(f)
         l = clandunit(c)
         g = cgridcell(c)
           .
           .
           .
         if (itype(l) == istwet .and. t_grnd(c) > tfrz) then
            h2osno(c) = 0._r8
            snowdp(c) = 0._r8
            snowage(c) = 0._r8
         end if
           .
           .
           .
      end do
```

Forrest Hoffman (forrest@climate.ornl.gov)
Mariana Vertenstein (mvertens@ucar.edu)

http://climate.ornl.gov/clm/vector_performance/

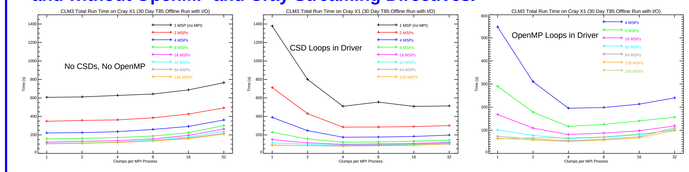
CLM3.0 Model Release Website
http://www.cgd.ucar.edu/tss/clm/distribution/clm3.0/

## Vectorization Process and Results

- Approval was obtained for the data structure modifications and vectorization strategy from the Land Model Working Group.
- Code was changed one subroutine branch at a time, testing answers and performance on the Cray X1 and IBM Power 4 along the way.
- Prototype code was provided to CREIPI and NEC for testing on the Earth Simulator and NEC SX platforms.
- Preliminary vectorization of CLM was completed ahead of the 2003 end−of−year deadline resulting in a single code that gives acceptable performance on both scalar and vector platforms.
- In October 2003, the new code was 25.8 times faster on the Cray X1 and was even 1.8 times faster on the IBM.
- CLM3.0 has a smaller memory footprint, and the new vector data structures simplify history updates and reduce the complexity and number of MPI gathers and scatters.

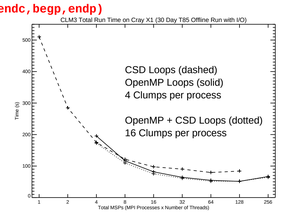
## Vector Performance on the Cray X1

- Timing tests were performed on the Cray X1 using the offline version of CLM3.0 varying processor count and the clumps−per−process parameter with and without OpenMP and Cray Streaming Directives.

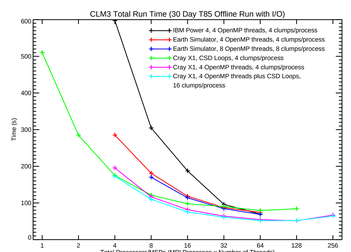



- The best performance is obtained when vector length is maximized. For no CSDs and no OpenMP, the clumps−per−process should be set to one, and when using shared memory it should be set to the number of shared memory processors (i.e., four in the CSD and OpenMP cases on the Cray X1).
- Performance turns over after 128 MPI processes (i.e., 128 MSPs).
- Best performance is achieved using OpenMP since lower−level loops are automatically multi−streamed by the Cray compiler.
- To test the performance of simultaneous use of OpenMP and Cray Streaming Directives at the driver routine level, the loops were split up as follows.

```
  nclumps = get_proc_clumps()
!$OMP PARALLEL DO PRIVATE (gnc,nc_beg,nc_end,nc,begg,endg,begl,endl,begc,endc,begp,endp)
do gnc = 1,nclumps/4
  nc_beg = (gnc − 1) * 4 + 1
  nc_end = min(nc_beg+3,nclumps)
!CSD$ PARALLEL DO PRIVATE (nc,begg,endg,begl,endl,begc,endc,begp,endp)
  do nc=nc_beg,nc_end
    call get_clump_bounds(nc, begg, endg, &
         begl, endl, begc, endc, &
         begp, endp)
      .
      .
    call Hydrology1(begc, endc, begp, endp, &
         filter(nc)%num_nolakec, filter(nc)%nolakec, &
         filter(nc)%num_nolakep, filter(nc)%nolakep)
      .
      .
  end do
!CSD$ END PARALLEL DO
end do
!$OMP END PARALLEL DO
```




- For this test, the clumps−per−process parameter was set to 16 (i.e., 4 for the SSPs times 4 for the OpenMP shared memory MSPs).
- Beyond 8 MSPs, no significant gain in performance was obtained by forcing simultaneous use of OpenMP and Cray Streaming Directives since most of the lower−level loops automatically multi−stream.




- Model performance was compared on the IBM Power 4, the Cray X1, and the Earth Simulator.
- At 64 processors, the Earth Simulator does slightly better than the Cray X1 using only CSDs.
- When OpenMP is used (with or without CSDs), the Cray X1 beats the Earth Simulator on a per processor basis.
- The Cray X1 offers significantly better performance than other platforms at low processor counts.

### ACKNOWLEDGEMENTS

This research used resources of the Center for Computational Sciences at Oak Ridge National Laboratory which is supported by the Office of Science of the U.S. Department of Enegy under Contract No. DE−AC05−00OR22725. Additional resources were provided by the National Center for Atmospheric Research which is sponsored by the National Science Foundation (NSF), the Central Research Institute of Electric Power Industry (CRIEPI) in Japan, Cray Inc., and NEC Corporation.