Recent Performance Improvements in the Community Atmospheric Model (CAM2)

P. H. Worley, F. M. Hoffman, M. Vertenstein, and J. B. Drake*

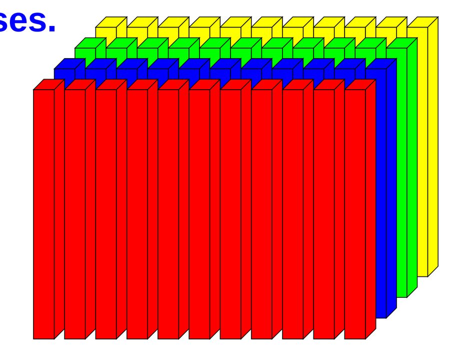
*Oak Ridge National Laboratory and *National Center for Atmospheric Research*

Introduction

One of the goals of the SciDAC project ‘‘Collaborative Design and Development of the Community Climate System Model for Terascale Computers’’ is to improve the performance portability of the Community Atmospheric Model (CAM2) and the Community Land Model (CLM2). Such improvements take two forms: 1) modifications that improve performance in all configurations on all platforms, and 2) compile− or run−time options that can be used to improve performance for a specific platform, processor count, or problem size. Since the inception of the project 18 months ago, significant progress has been made in both areas. Described here are performance impacts of improvements to the physical parameterizations and the spectral Eulerian dynamical core as well as the implementation of a new interface between CAM2 and CLM2. Performance results are shown for the IBM p690 cluster at Oak Ridge National Laboratory and the HP AlphaServer SC ES45 cluster at Pittsburgh Supercomputer Center.

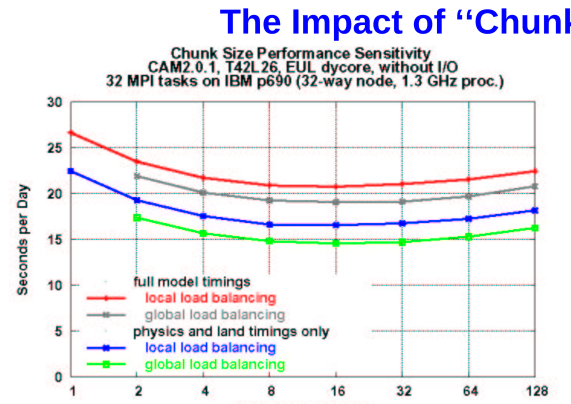
Physical Parameterization Optimizations

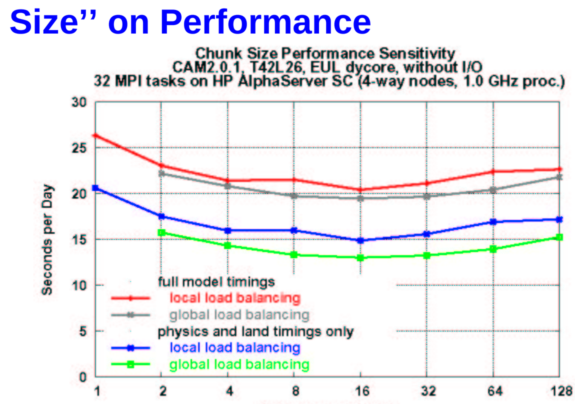
- Just prior to the SciDAC project, CAM developers chose to separate the data structures and parallel algorithms of the physical parameterizations (physics) from the dynamical core (dycore) to
  - support multiple dycores (spectral Eulerian, spectral semi−Lagrangian, and finite volume semi−Lagrangian) and
  - allow physics and dynamics parallel algorithms to be optimized separately.
- The SciDAC project introduced a new computational unit for the physics called the ‘‘chunk’’ −− an arbitrary subset of vertical columns distributed among MPI processes.
  - Two tuning parameters: number of columns assigned to a chunk and number of chunks assigned to MPI processes.
  - Each MPI process is assigned at least one chunk, and OpenMP parallelism is exploited when more than one chunk is assigned to an MPI process.
  - Two decomposition strategies:
    1. Use the same parallel decomposition as in the dycore, avoiding interprocess communication between the physics and dycore. The chunks are defined to balance the work associated with each chunk for a given process, thus load balancing any OpenMP parallelism within an MPI process.
    2. Assign columns to chunks to balance the load across all chunks, and assign the same number of chunks to each processor. Chunk assignment also attempts to minimize the interprocessor communication in the physics/dycore interface.
- The chunk size (i.e., number of columns per chunk) determines memory access patterns, and can be tuned to the cache size or vector length of the computer architecture.



Vertical columns in the atmosphere arranged into ‘‘chunks’’ for parallel decomposition and load balancing.

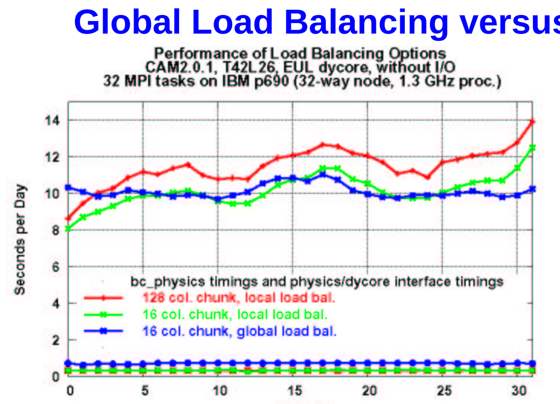
The Impact of ‘‘Chunk Size’’ on Performance

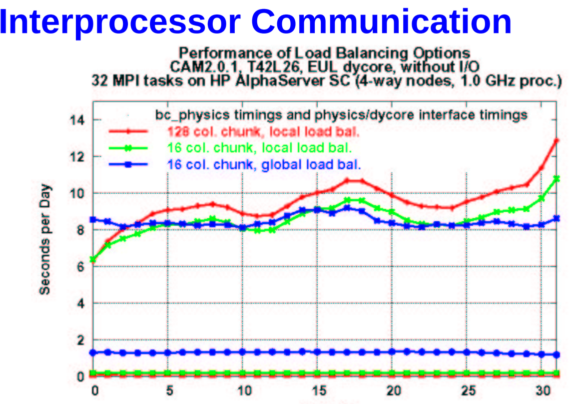




The performance of CAM on a 128x64 horizontal grid with 26 vertical levels (T42L26) using 32 MPI processes on a 32−way IBM p690 node (left) and on an HP AlphaServer SC (eight 4−way ES45 nodes) (right) is plotted as a function of the number of columns per chunk. Results are given for both globally load balanced chunks (option 1 above) and locally load balanced chunks (option 2 above). The locally load balanced 128−column chunks are equivalent to the latitude−slice decomposition formerly used in CAM. Neither small nor large chunks are optimal, and global load balancing improves performance. In addition, performance variation due to chunk size is identical to that of just the physics and the land model. (The land timings can not be separated from the physics timings due to load imbalances.)

Global Load Balancing versus Interprocessor Communication

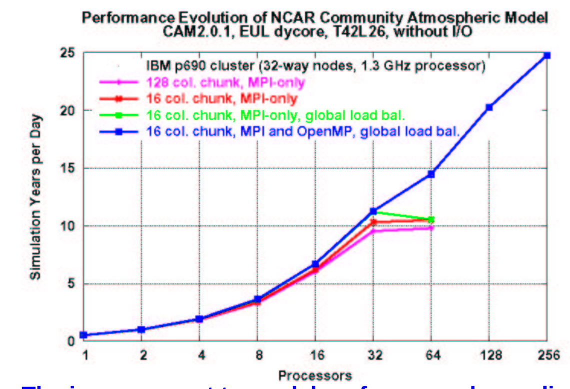


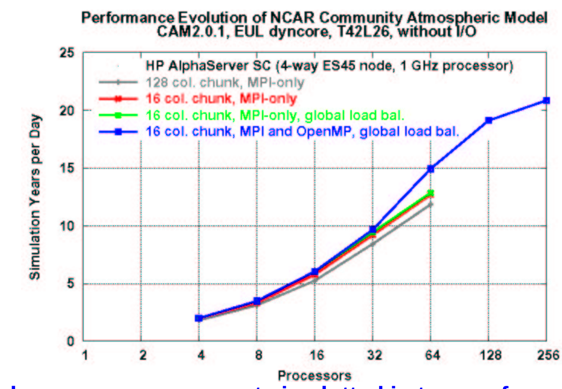


In these figures, the time spent in the bc_physics routine is plotted (top curves) along with the time spent moving data between the physics and the dycore, which includes any necessary inter−processor communication (bottom curves), for each of 32 MPI processes on the IBM p690 (left) and the HP AlphaServer SC (right). Time spent in bc_physics is 90% of the time spent in physics and represents all of the physics load imbalance. While the global load balancing algorithm is not perfect, it decreases the range of variability by more than a factor of four in these experiments. The additional time spent moving data when using the global load balancing algorithm (bottom curves) is measurable, but the improvement due to global load balancing greatly outweighs this cost. This difference is a function of interprocessor communication performance, and a different result may hold on another architecture or for a different processor count or problem size.

Scaling Physics to Many Processors

Decoupling of the physics and dynamics data structures and parallel algorithms allows the physics to use more processors than the spectral Eulerian and spectral semi−Lagrangian dycores. These dycores presently use a one−dimensional decomposition, and consequently can use at most 64 processors at a T42 horizontal resolution. Moreover, scalability of the dycores is poor when increasing from 32 to 64 processors. In contrast, the physics can now use up to 8192 processors for the same problem size. One approach to improving scalability is to use the same number of MPI processes in the physics and the dycore, but use additional processors in the physics via OpenMP parallelism.





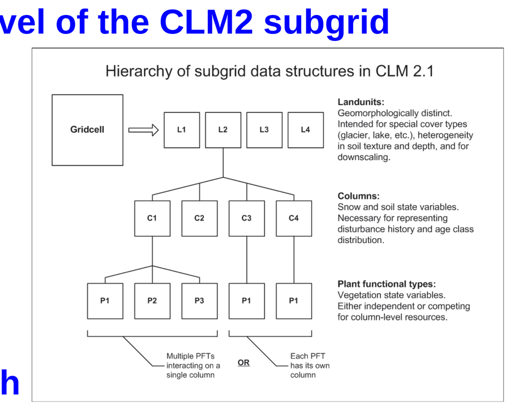
The improvement to model performance by scaling to larger processor counts is plotted in terms of simulation years per day of computation for the IBM p690 (left) and the HP AlphaServer SC (right). For the mixed MPI/OpenMP experiments, the optimal mix of MPI processes and OpenMP threads was determined for each total processor count. While the chunk size and load balancing optimizations improve performance by approximately 30% for 32 processors, the major benefit is in increased scalability when using OpenMP and more than 32 processors.

Spectral Dycore Optimizations

- To further improve scalability, work is underway to implement a two−dimensional parallel decomposition into the spectral Eulerian and spectral semi−Lagrangian dycores which presently use a one−dimensional decomposition.
- Prior to adding this two−dimensional decomposition, the current decomposition was optimized by
  - implementing a consistent one−dimensional decomposition in the spectral data structures,
  - redefining the decomposition to improve load balancing, and
  - eliminating unnecessary interprocessor communication.
- The overall impact was a significant decrease in communication overhead, decreasing run time on all systems.

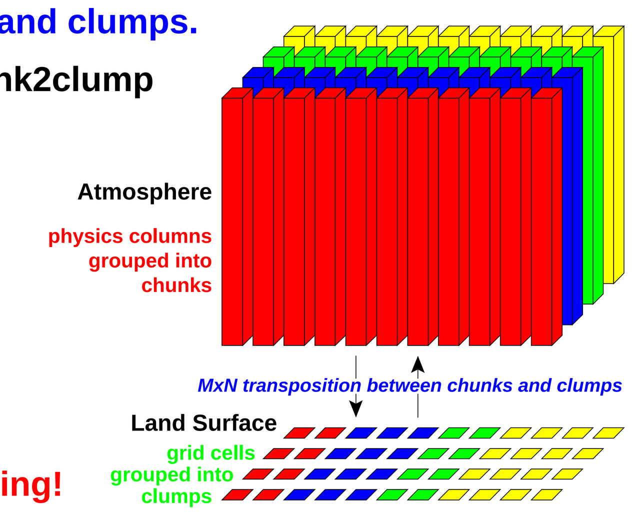
A New Atmosphere/Land Interface

- The Community Land Model (CLM2) simulates land surface processes on a hierarchy of specialized subgrid patches.
- Land surface grid cells, the highest level of the CLM2 subgrid hierarchy, correspond to atmosphere columns when run with CAM2.
- The SciDAC project introduced a new computational unit for the land called the ‘‘clump’’ −− an ordered subset of grid cells distributed among MPI processes.
  - The computational workload for each grid cell is proportional to the number of subgrid patches (plant functional types) contained within the grid cell. To balance the work across all clumps, the clumps consist of a varying number of grid cells.
  - A grid cell may not span multiple clumps.
  - Two decomposition strategies:
    1. One clump is created for each MPI process, and the patches are (nearly) evenly divided among clumps by way of whole grid cells. This is the default decomposition.
    2. If a suggested number of patches per clump is specified, the grid will be decomposed into as many clumps as necessary containing at least that number of patches in terms of whole grid cells. Clumps are then evenly divided among MPI processes.



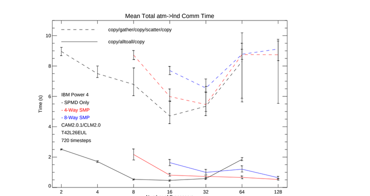
Grid cells on the land surface arranged into ‘‘clumps’’ for parallel decomposition and load balancing.

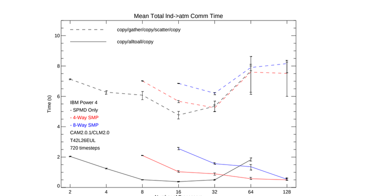
- CLM2 is usually run coupled directly with CAM2.
- The models run in parallel as a single executable, share the same processors, and exchange surface fluxes and states via MPI.
- A new interface between the atmosphere and the land surface was implemented to provide an efficient MxN transposition between physics chunks and land clumps.
- Two new data structures, chunk2clump and clump2chunk provide the mapping between the two.
- lp_coupling_init() initializes these data structures, and it may be called repeatedly to re−initialize the mapping if the decomposition of either the physics or the land changes. Supports dynamic load balancing!
- Atmosphere columns over the ocean are ignored.
- Multiple MPI gathers and scatters were replaced by a single MPI_alltoallv() before and after the call to the land driver() routine.
- After data structures are initialized, no additional computation is required to perform data exchange via MPI.



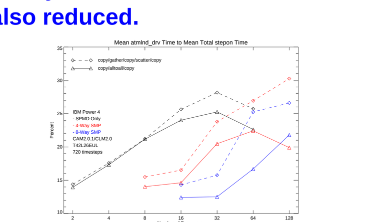
CAM2/CLM2 Interface Performance

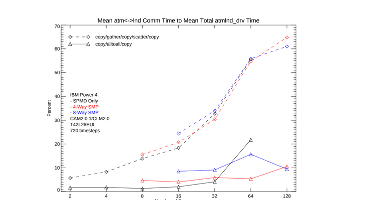
To gauge performance of the new atmosphere/land interface, tests were run on the IBM Power 4 (p690) at Oak Ridge using CAM2.0.1 with CLM2.0 at T42L26 with the spectral Eulerian dycore.





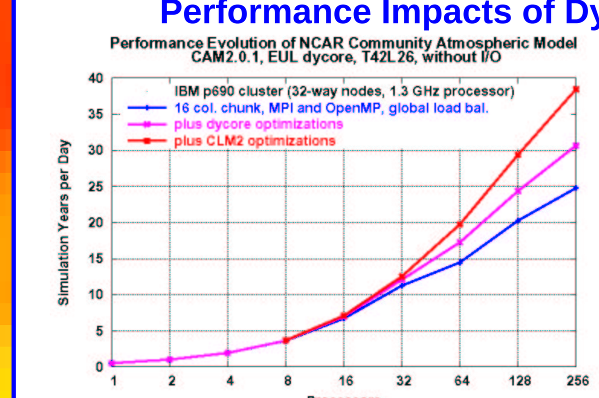
The left figure shows mean communications time from the atmosphere to the land while the right figure shows the mean communications time from the land to the atmosphere. Dashed curves represent the old interface; solid curves represent the new interface. Mean total communications time is significantly reduced in the new interface, and the variance of communications times across processes is also reduced.

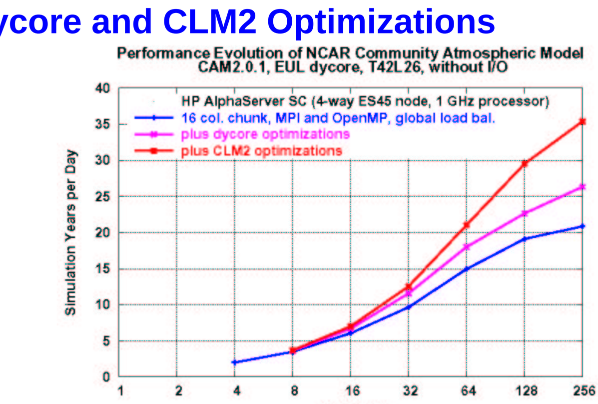




The left figure shows the ratio of mean land model (CLM2) run time to mean full model (CAM2+CLM2) run time. Dashed curves represent the old interface; solid curves represent the new interface. Since communication time is attributed to the land model, the improved interface reduces total land model run time so that CLM2 represents a smaller percentage of total run time. The right figure shows the ratio of mean communications time to mean total land model run time. The new interface significantly reduces the percentage of total land model run time consumed by communications.

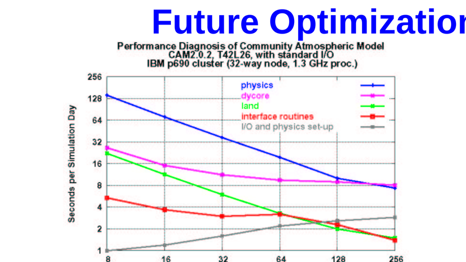
Performance Impacts of Dycore and CLM2 Optimizations

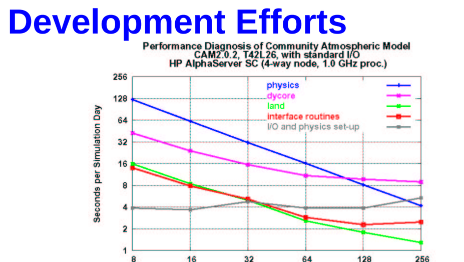




The performance impacts of the dycore optimization and the new land model interface are plotted in terms of computational rate. While there is some performance improvement for all processor counts, as with earlier optimizations, the biggest gains are in scalability for large processor counts. This indicates that CAM was communication bound for processor counts greater than 32 before these two optimizations were introduced.

Future Optimization and Development Efforts





Timings for physics, Eulerian dycore, land model, physics/dycore and CAM2/CLM2 interface routines, and I/O & physics set−up are shown separately above. The interface routines are dominated by inter−processor communication and the I/O & physics set−up are dominated by I/O and serial bottlenecks.

- Up to 32 processors, physics dominates the model performance. Improving the serial performance of the physics could improve model performance on both the IBM and HP systems. However, there is also a performance anomaly on the IBM when moving from 128 to 256 processors.
- For more than 32 processors, the dycore begins to limit CAM2 performance. Dycore performance stops improving when using more than 64 processors. Implementing a two−dimensional domain decomposition in the dycore should address this problem.
- While the land scales well enough not to limit the performance of CAM2, additional optimizations will be made to improve the performance and scalability of CLM2.
- Interface routines scale reasonably well on the IBM. When the new IBM Federation switch is installed, the interface routines should become even less of a performance factor. On the HP, the performance of the interface routines is a problem. This may be attributable to an inefficient implementation of the MPI_Alltoallv() collective routine. An alternative implementation will be examined in an attempt to address this problem on the HP.
- The I/O & physics set−up routines both have significant serial bottlenecks. A new parallel I/O library is being developed, and the serial bottleneck in the physics set−up will also be eliminated in a future modification. Note that the performance on the HP is also affected by the use of other MPI collective routines that appear to perform poorly. Alternative implementations of these routines will also be examined.

SciDAC and Biology & Environment

United States Department of Energy Oak Ridge National Laboratory

UT-BATTELLE
Management Contractor for DOE's Oak Ridge National Laboratory