



## Parallel Analysis Tools and New Visualization Techniques for Ultra-Large Climate Data Sets (ParVis)

### PROGRESS REPORT: 2011-2012

Argonne National Laboratory: Robert Jacob, Xiabing Xu, Jayesh Krishna, Sheri Mickelson, Tim Tautges, Mike Wilde, Robert Latham, Ian Foster, Rob Ross, Jay Larson  
Sandia National Laboratory: Pavel Bochev, Kara Peterson  
Pacific Northwest National Laboratory: Karen Schuchardt, Jian Yin, Tekin Bicer  
National Center for Atmospheric Research: Don Middleton, Mary Haley, David Brown, Richard Brownrigg, Wei Huang, Dennis Shea, Mariana Vertenstein  
University of California-Davis: Kwan-Liu Ma, Jinrong Xie

## Summary

ParVis is a project funded under LAB 10-05: “Earth System Modeling: Advanced Scientific Visualization of Ultra-Large Climate Data Sets”. Argonne is the lead lab with partners at PNNL, SNL, NCAR and UC-Davis.

This report covers progress from October 1<sup>st</sup>, 2011 through Dec 21<sup>st</sup>, 2012.

A primary focus of ParVis is introducing parallelism to climate model analysis to greatly reduce the time-to-visualization for ultra-large climate data sets. For this report, it is convenient to summarize the work as two tracks with different time horizons: one track is to provide immediate help to climate scientists already struggling to apply their analysis to existing large data sets. The other track focused on building a new data-parallel library and tool for climate analysis and visualization that will give the field a platform for performing analysis and visualization on ultra-large datasets for the foreseeable future.

***We made good progress on our second-year milestones and completed our 2<sup>nd</sup>-year deliverable: task parallel (with Swift) versions of other CESM diagnostic packages. We completed the ocean and sea-ice model working group packages to add to the atmosphere model working group package from the first year.***

***We also made a beta release of our ParNCL application to the community.***

## Progress on short-term improvements

### Swift-based climate model diagnostics

Swift is a system for the rapid and reliable specification, execution, and management of large-scale science and engineering workflows. It supports applications that execute many tasks coupled by disk-resident datasets - as is common, for example, when analyzing large quantities of data or performing parameter studies or ensemble simulations. The diagnostic plots made by c-shell scripts developed by CESM working groups can straightforwardly be recast as many-task parallel applications. The diagnostic scripts use a combination of NCO utilities (ncra, etc.) to perform data reduction and NCL to make additional calculations and plots. Plots from these scripts are used extensively by scientists evaluating climate model integrations.

*The introduction of task parallelism to diagnostic workflows has been very successful and provided immediate speed increases.*

A task parallel version of the Atmospheric Model Working Group diagnostics was completed in our first year and released on the ParVis home page. The Swift version was merged with the main development and released in version 5.3 of the AMWG diagnostics on January 31, 2012.

Since the release of our version of the Atmospheric Model Working Group diagnostics last year, we have updated the package to re-grid SE files to FV. This was needed to plot the results with NCL. We also were able to create a branch version that replaces the NCO calls with Pagoda. Pagoda, developed as part of the Colorado State University Global Cloud Resolving Model project, provides parallel versions (using Global Arrays) of the NCO utilities. This introduced a layer of data-parallelism to the already task-parallel version of the AMWG diagnostics package. We were able to run this version with 5 years of  $\frac{1}{4}$  degree SE data that was upsampled to  $\frac{1}{10}$ th degree FV grid. Each of the monthly history files were 24GB. The results are shown in Figure 1. The Swift-only version was able to compute the climate averages 4X faster. Running with pagoda instead on NCO, the Swift/pagoda version was able to run in less than 1 hour and cut the Swift-only time by about half. The original serial version took almost 9  $\frac{1}{2}$  hours to compute the same files.

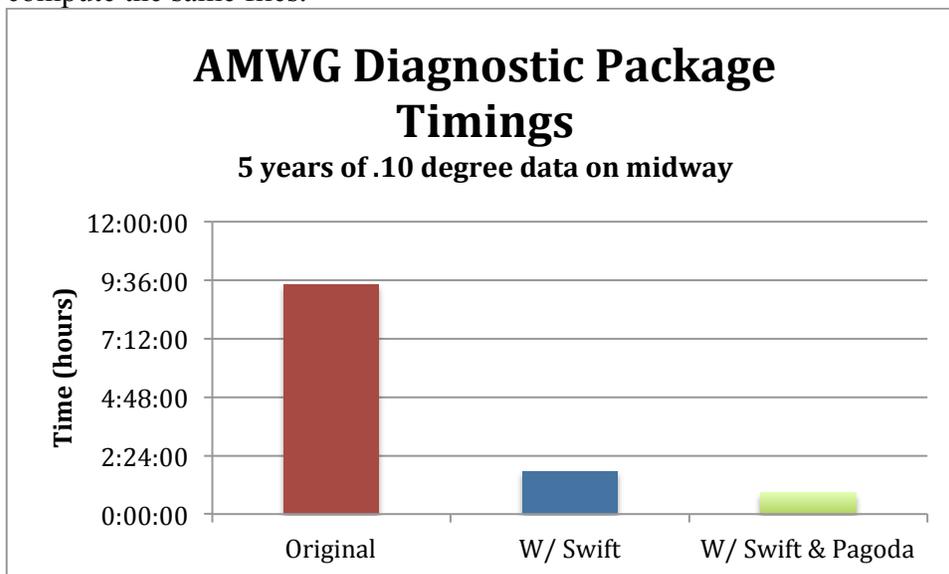


Figure 1: Swift speedups for high-resolution atmosphere data using Swift and Pagoda

Since the last report we've also released an NCL and Swift version of the Ocean Model Working Group (OMWG) diagnostic package. The original version of the package used IDL to create the plots. To achieve more portability, the NCAR NCL team rewrote the original IDL scripts in NCL. We also re-wrote the top level c-shell scripts in Swift to compute the climate average files, create the NCL plots, and convert the images files in parallel. The results are shown in Figures 2-4. The OMWG diagnostic package contains three separate scripts. The first script compares model data to observational data (Figure 2). The second compares two model runs against each other (Figure 3). The final script computes a time series (Figure 4).

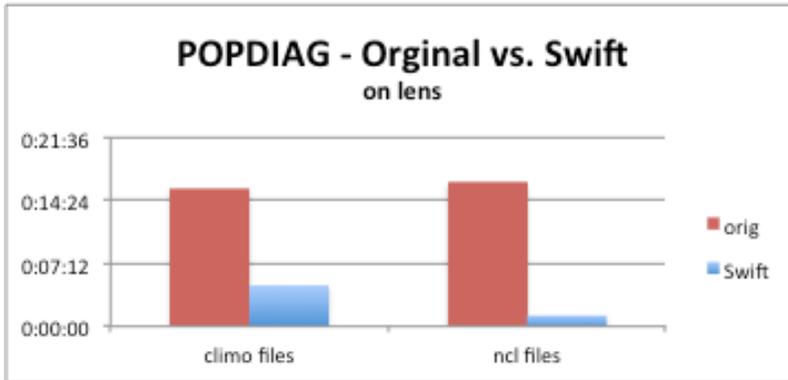


Figure 2: Timings for the OWMG diagnostic package popdiag script

All of the timings were run using the same 10 year 1 degree history files. The popdiag script compared the 10 years against observational data. The popdiag script compared the first 5 years against the later 5 years. The popdiagts script

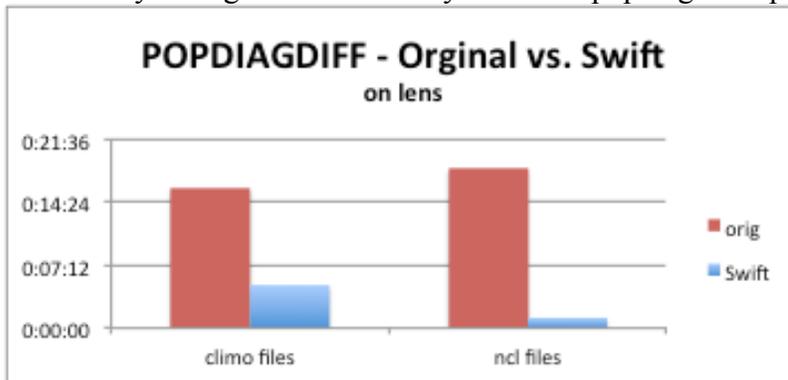


Figure 3: Timings for the OWMG diagnostic package popdiagdiff script

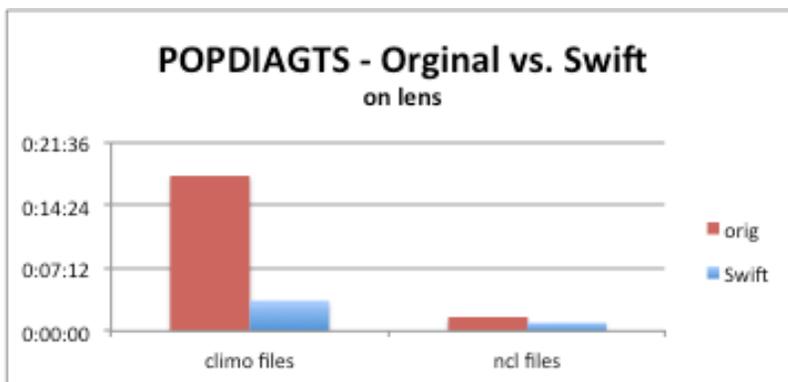


Figure 4: The OWMG diagnostic package popdiagts script

created a time series for the full 10 years of data. The popdiag and popdiagdiff scripts each saw about a 3x speedup computing the climate average files and saw a very large improvement in the time it took to create the plots using NCL. The popdiagts script saw an even greater improvement in computing the climate files because there were more tasks that were able to be computed in parallel. There wasn't as big of an improvement seen with the NCL plots because there aren't as many scripts available for task

parallelism. This version was released to the community through the OMWG in June, 2012.

Since this first release of the ocean model diagnostics, we've released a second version that allows users to run with hi-resolution data (1/10<sup>th</sup> degree). We tested this second version comparing 1 year of 1/10<sup>th</sup> degree data against observations. The results are shown in Figure 5. The original version took about 6 ¾ hours to run. The Swift version was able to reduce this time to about 1 ½ hours.

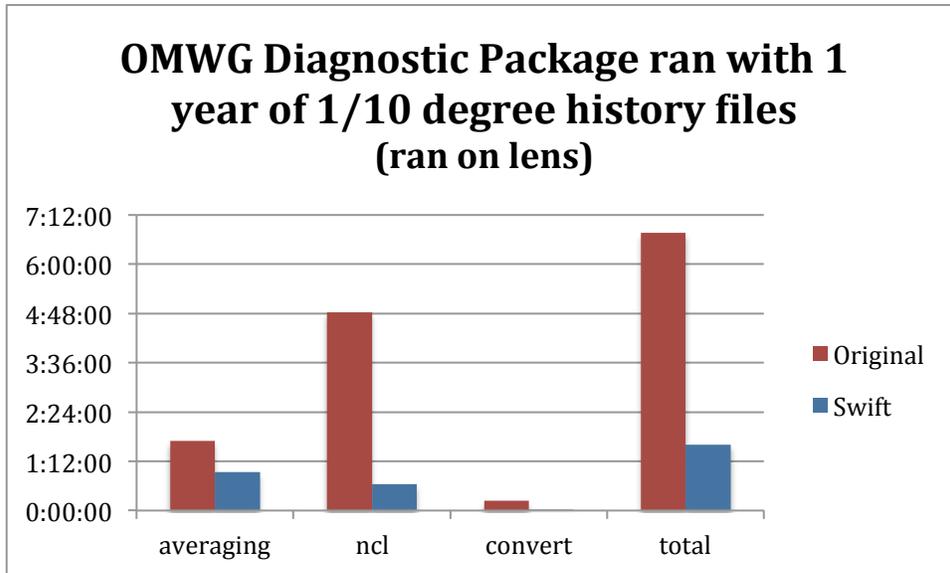


Figure 5: The OMWG diagnostic package ran with hi-resolution data

We have also complete work on re-writing the Ice Model Diagnostic Package in Swift. It's currently available to friendly users and we hope to release it to the community soon. Our initial timing results show almost a 4x speedup. Figure 6 shows the results of comparing 2, 20 year 2 degree data sets to each other.

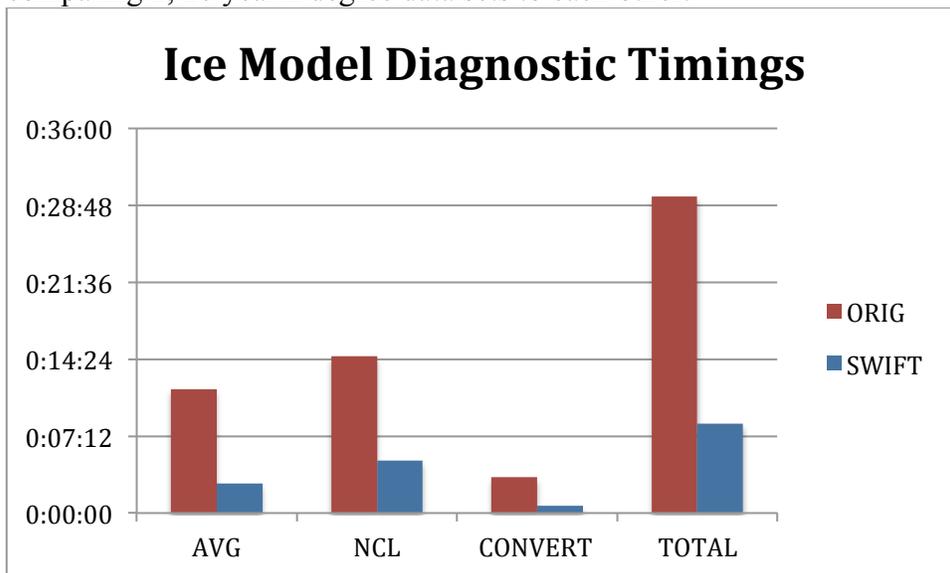


Figure 6: The Ice Model Diagnostic Package timings comparing two datasets to each other

We have also started work on the Land Model diagnostic package. We hope to have this completed in the first quarter next year. After this work is complete, we will focus on adding Pagoda to all four diagnostic packages.

## Improvements to NCL

The NCL team completed the integration of Earth System Modeling Framework (ESMF) parallel regridding software into a May 2012 release of NCL. The ESMF software allows users to regrid to and from various topographically rectangular and is critical in the IPCC AR5 for comparing model runs generated on large, complex, and dissimilar grids.

The NCL profiling capability developed last year was included in the May 2012 beta release and the 6.1.0 release in October, 2012.

The NCL team began an effort to modernize the graphics code with a greater focus on performance and more accurate rendering of both structured and unstructured grids. A new MeshFill plotting algorithm is being developed to explicitly render arbitrary polygonal mesh cells to a raster plot. Benchmarking a preliminary version of the MeshFill method indicates that it offers a significant speed-up comparing with existing fill plotting. A beta version of this code was made available in the version 6.1.0 release of NCL in October, 2012. Team members are actively exploring opportunities to add parallelism to help speed up the code for fill plotting as well as for generic polyline and polygon rendering.

In the summer of 2012, the NCL team sponsored a SIParCS project to develop a Python-based web interface that employs task parallelism to facilitate dynamic comparison of ocean model output. The outcome of this project will be presented at a Python symposium at the 93rd AMS Annual Meeting in January 2013.

## Progress on long-term tasks

ParVis' long term plan for enabling ultra-large climate data analysis involves developing a new high-performance data-parallel library for performing standard climate calculations on both regular and unstructured grids, the Parallel Grided Analysis Library (ParGAL, formerly called the Parallel Climate Analysis Library). ParGAL is used to build a parallel version of NCL called ParNCL. We are also performing working to take in to account future hardware considerations including exploring compression, cloud computing approaches to analysis and new approaches to 3D visualization of climate data. Significant progress was made on all tasks.

### ParGAL development

After completing a working prototype in the first year, we now have a released version of ParGAL. ParGAL is built upon the Mesh Oriented database (MOAB), Parallel Netcdf (PnetCDF) and Intrepid libraries. The combination of MOAB and Intrepid enables the concise and succinct expression of the complex and computationally intensive algorithms such as computing vorticity and divergence.

In the past 15 months, the library has been expanded to support the reading and analysis of more types of grids. ParGAL must take care in reading gridded data because the algorithms use knowledge of the discretization, knowledge which is usually not included in the NetCDF metadata and so must be added by ParGAL/MOAB developers. Currently we are able to handle data on three grid types from the Community Atmosphere Model, the Eulerian Spectral grid (CAM-Eul), Finite Volume (CAM-FV) grid and, most significantly, the unstructured Spectral Element (CAM-SE or HOMME) grid.

### **ParGAL's Pcvr class updates**

The Pcvr class has been updated to support variables read from file(s) or created by the user on all of three CAM grids (see MOAB section below). The types of grids are transparent to the user, so they don't have to be concerned about the underlying implementation. However, the user could get the grid type information by issuing a function call `get_grid_type` on a `fileinfo` object that stores the information about the file(s) read in.

The user can still create a Pcvr by a name represented with a string and an argument stating whether it is read from file or user created as before. Beyond that functionality, the Pcvr object can also be built by taking user created MOAB tags. This is a feature intended for advanced users since the memory allocation and deallocation must be controlled explicitly by the user.

ParGAL's Pcvr class now supports missing data. If the Pcvr object is created by reading a file variable, then an attribute called `missing value` or `fill value` will be checked. If such an attribute exists, then the missing value data member will be set for the Pcvr object. If the Pcvr object is created by the user, then the user has the option to specify a missing value. If the file variable doesn't have the missing value attribute or the user hasn't specified the missing value, then it will be set to 0 by default. Operations on a missing value will be ignored when the Pcvr object is used to perform various data analysis.

Depending on the different types of the grid and the discretization, the variables could be stored at different locations on a MOAB mesh instance, such as on the file set, on the nodes, on the edges or on the cell centers. The location information will be set automatically for variables read from the file. The user has the freedom to set the location of user created Pcvr. It will be initialized to cell centers by default if the user doesn't specify one. Pcvr continues to provide the abstraction of the data stored on the MOAB mesh instance. The user doesn't have to have the knowledge of where the data is stored.

## **ParGAL algorithm updates**

Three of the most time-consuming algorithms and the gather algorithm have been re-implemented based on MOAB's new 2.5D representation of the structured grid (more information below), including `dim_avg_n`, `max` and `min` (changed to the same name used in NCL). Two of the new algorithms `dim_max_n` and `dim_min_n` are added. All of these algorithms will work for any dimension of the Eulerian Spectral grid and Finite Volume grid data. It also works for the HOMME grid if the input dimension is either time or level. The new versions of the algorithms are also now able to handle missing values. The algorithms won't distinguish the Pcvr constructed by different scenarios, file variables or user created, the types of grids the variable is from etc.

## **ParGAL Software Engineering**

We are following modern software engineering practices in developing ParCAL. Autotools (`autoconf`, `automake` and `libtool`) have been used for generating automatic configuration scripts. “`make`”, “`make install`” and “`make check`” are all implemented. “`make check`” invokes unit tests created with the Boost Test Library - Unit Test Framework. The nightly build/test system has been expanded to test in 11 different configurations, and results are now open to the public. Continuing to follow good software development practice, all of the warnings generated by `gcc` are removed. The detailed build instructions and doxygen generated documentation could be found on the ParVis wiki. The library has also been tested on different platforms (Linux, MacOS) using different compilers (`gcc`, intel and `pgi` compilers).

## **MOAB/ParGAL development**

We have made several changes to MOAB to facilitate the development of ParGAL.

The representation of structured grids in MOAB was changed from a fully 3D representation to a 2.5D representation to reduce the memory footprint; each variable has an extra dimension represented by the number of levels or intermediate levels, corresponding to the altitude. This has an implication on data ordering, as each level is contiguous in the file, so a transpose is necessary right after reading. This change also required re-writing many of the ParGAL algorithms already implemented.

We have also enabled MOAB to better represent structured periodic grids in parallel and serial.

One of the unique features of ParGAL through MOAB is the ability to represent the model output data on the same discretization as was used in the model while computing the solution. In our initial prototype, we were treating all variables as if they were on the nodes of a computational grid. In reality, most quantities are computed on cell centers. We made substantial changes to the NetCDF reader in MOAB to distinguish between different grid formats so that variables were read into MOAB and associated with the appropriate mesh geometrical entity, for example temperature at the element center rather than an element node. Since this information is not in the NetCDF file, we added logic to

the MOAB NetCDF reader to recognize which dycore was used (Eul, FV or SE) and place values on the appropriate entities. Similar logic will have to be added for other grid types while we develop a more general solution.

Other data in the NetCDF file is also now read in the MOAB. Global attributes in the file are saved as sparse tags on the local mesh set on each processor. Variable data are saved as dense tags associated to the nodes and elements in the mesh. Direct access to the memory occupied by the data is provided for dense tags, so the client can avoid expensive copy after the data has been read from disk.

Additional changes in the specialized MOAB NetCDF reader were made for structured and unstructured meshes.

1) structured mesh

- Improve the performance of structured mesh interface, for mesh and variable reading in parallel;
- specialize sharing and ghosting for structured grids.
- Allow and identify periodic boundaries in the data automatically

2) unstructured mesh CAM-SE (HOMME grids)

- implement trivial partition, in which each processor gets a balanced number of quad elements, ordered by their global index.
- resolve sharing and ghosting using regular MOAB parallel communicator functions, for unstructured meshes
- read nodal variables using asynchronous pnetcdf calls, on each processor, for each variable. Each processor reads fragments of the data corresponding to the local represented nodes, only.
- introduce a global mesh set on the root processor, that can be used to gather a variable (tag) from domain-decomposed meshes onto the root;
- introduce the spectral mesh tool

**Near Term development.**

We plan to complete the implementation of spectral mesh tool to allow for a direct use of Intrepid spectral element functions and methods for various algorithms. For structured meshes, allow certain data to be associated with edges. We will work with CAM developers to eliminate the need for the extra connectivity array for HOMME grids; the connectivity could be included in the regular data file. We will extend the reader to other climate specific meshes, like MPAS grids. Finally we will develop a dedicated partitioner for better balancing for unstructured meshes and extend the ghosting capability.

**Intrepid/ParGAL development**

The work of Sandia's team during this reporting period focused on the development, testing, and implementation of parallel algorithms to perform operations currently

handled by NCL spherical harmonic functions. The ParGAL implementation uses the finite element method to compute derivatives for quantities such as vorticity and divergence and to solve simple partial differential equations for streamfunction and velocity potential.

Initial implementation of the ParGAL functions provided a capability to compute these quantities on quadrilateral elements for given nodal velocity components. As mentioned above, additions to the MOAB netcdf file reader provided the ability to distinguish between different grid formats so that variables were read into MOAB and associated with the appropriate mesh geometrical entity. This led to the need for alternative formulations of the algorithms for velocity components given at locations other than element nodes that we implemented in the past year.

Specific grid-based formulations have been developed for the three CAM grids ParGAL and MOAB can currently read. Output for CAM-Eul and CAM-FV contain velocity components at the cell centers and for CAM-SE the velocity components are located on the cell nodes. Progress on algorithms for each of the grids is given in more detail in the sections that follow. In addition, Table 1 provides a description of the NCL spherical harmonic functions that have been adapted along with the status of the parallel algorithm development.

Table 1: Status of development and implementation of parallel algorithms for the NCL spherical harmonic functions.

<b>NCL Function</b>	<b>Description</b>	<b>Status</b>
<b>uv2dv(F,f,G,g)</b>	Divergence from velocity field	In ParGAL for CAM-FV, CAM-Eul and CAM-SE grids
<b>uv2vr(F,f,G,g)</b>	Vorticity from velocity field	In ParGAL for CAM-FV, CAM-Eul and CAM-SE grids
<b>uv2vrdiv(F,f,G,g)</b>	Vorticity and divergence from velocity field	In ParGAL for CAM-FV, CAM-Eul and CAM-SE grids
<b>uv2sfvp(F,f,G,f)</b>	Streamfunction and velocity potential from velocity field	Implementation developed for nodal quantities, structured grids
<b>sfvp2uv(f,g)</b>	Wind components from streamfunction and velocity potential	Implementation developed for nodal quantities, structured grids

### **Vorticity/divergence for CAM-Eul and CAM-FV Grids**

Algorithms were developed and tested to compute vorticity and divergence on structured Gaussian or fixed latitude/longitude grids given cell-centered velocity components. The vorticity and divergence are computed with the finite element method, which provides a method to handle unstructured meshes and is straightforward to parallelize. A formal L2 projection is used to build a linear system that is solved for the cell-centered vorticity or divergence values. The algorithm leverages several packages from the Trilinos project including Intrepid for element basis functions and quadrature rules, Epetra for distributed linear algebra tools, and AztecOO and ML for solving and preconditioning the linear system.

For testing purposes a velocity field derived from a streamfunction that is a wavenumber four Rossby-Haurwitz wave and a low-order spherical harmonic function for the velocity potential has been used. Given this velocity field an exact expression for the vorticity and divergence can be derived and errors between the exact solution and our finite element approach calculated.

Convergence rates for the vorticity and divergence algorithms were determined by computing the Rossby wave solution on two sets of grids. The first set includes three grids in a CAM-FV configuration that are equispaced and have resolutions of approximately 4 degrees (48x72), 2 degrees (96x144), and 1 degree (192x288). The second set consists of the three Gaussian grids T31, T42, and T85 with resolutions of 48x96, 64x128 and 128x256, respectively. Results from the convergence test are given in Tables 2 and 3. Plots of the vorticity and divergence for the test case from the

Table 2: Errors and convergence rates in the ParGAL computation of vorticity for the Rossby wave test case.

<b>Mesh</b>	<b>Size</b>	<b>L2 error</b>	<b>Rate</b>	<b>L1 error</b>	<b>Rate</b>
<b>4° FV</b>	48x72	$4.45 \times 10^{-3}$		$2.18 \times 10^{-3}$	
<b>2° FV</b>	96x144	$1.35 \times 10^{-3}$	1.72	$5.51 \times 10^{-4}$	1.98
<b>1° FV</b>	192x288	$4.47 \times 10^{-4}$	1.66	$1.42 \times 10^{-4}$	1.97
<b>T31</b>	48x96	$1.58 \times 10^{-3}$		$1.26 \times 10^{-3}$	
<b>T42</b>	64x128	$9.34 \times 10^{-4}$	1.83	$7.36 \times 10^{-4}$	1.87
<b>T85</b>	128x256	$2.51 \times 10^{-4}$	1.84	$1.98 \times 10^{-4}$	1.89

Table 3: Errors and convergence rates in the ParGAL computation of divergence for the Rossby wave test case.

<b>Mesh</b>	<b>Size</b>	<b>L2 error</b>	<b>Rate</b>	<b>L1 error</b>	<b>Rate</b>
<b>4° FV</b>	48x72	$1.11 \times 10^{-2}$		$5.37 \times 10^{-3}$	
<b>2° FV</b>	96x144	$3.37 \times 10^{-3}$	1.72	$1.33 \times 10^{-4}$	1.99
<b>1° FV</b>	192x288	$1.12 \times 10^{-3}$	1.65	$3.45 \times 10^{-5}$	1.97
<b>T31</b>	48x96	$7.77 \times 10^{-3}$		$4.30 \times 10^{-3}$	
<b>T42</b>	64x128	$4.87 \times 10^{-3}$	1.62	$2.51 \times 10^{-4}$	1.87
<b>T85</b>	128x256	$1.71 \times 10^{-3}$	1.54	$6.67 \times 10^{-4}$	1.90

ParGAL finite element implementation and the standard NCL spherical harmonic implementation for the T42 grid are shown in Figure 7.

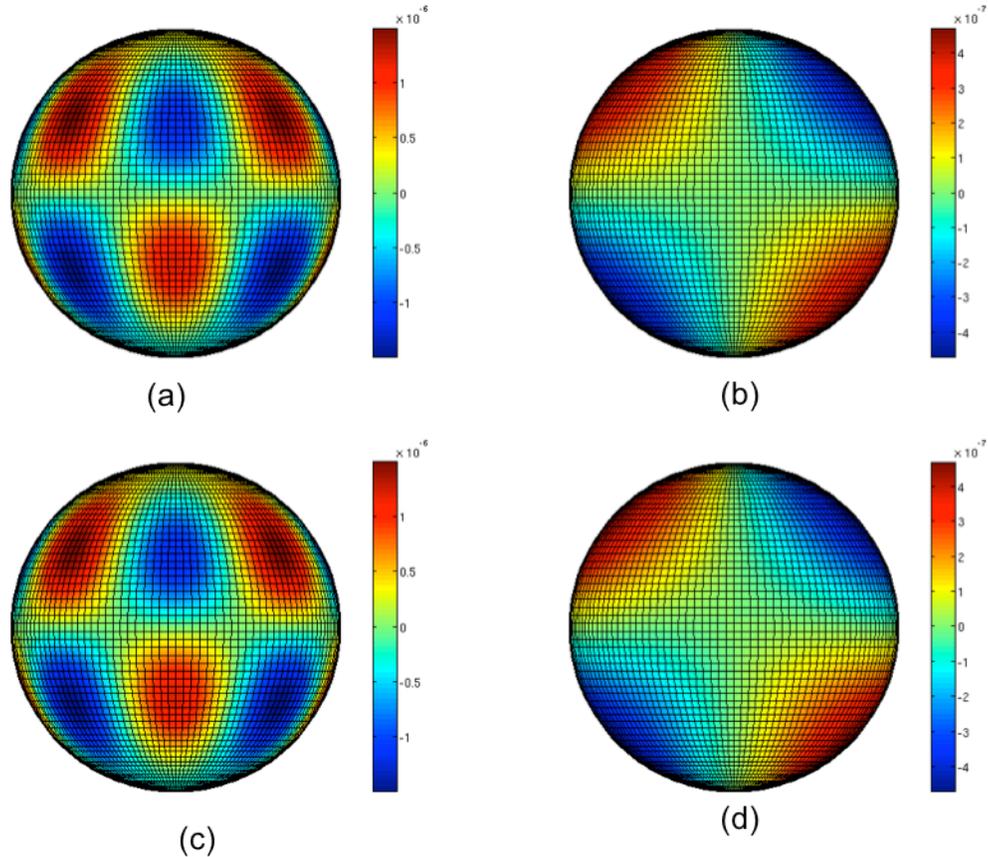


Figure 7. Vorticity(a) and divergence(b) from the ParGAL algorithm and vorticity(c) and divergence(d) from the NCL spherical harmonic algorithm on the T42 CAM-Eul grid for the Rossby wave test case.

### Streamfunction and Velocity Potential for CAM-Eul and CAM-FV Grids.

Algorithms were developed to compute the streamfunction and velocity potential given nodal velocities on structured Gaussian or fixed grids. Work continues on implementing the algorithms for cell-centered velocities. The streamfunction and velocity potential are the solutions of the Poisson equations. The finite element method is used to solve these equations using a weak formulation that depends explicitly on the velocity components rather than the divergence and vorticity.

To test the algorithm the streamfunction and velocity potential were computed for the Rossby wave velocity field on grids from the two mesh sets described in the previous section. Convergence rates are shown in Tables 4 and 5. Plots of the streamfunction and velocity potential for the test case from the ParGAL finite element implementation and the standard NCL spherical harmonic implementation for the T42 grid are shown in Figure 8.

### Vorticity/divergence for CAM-SE Grid.

Algorithms were developed to compute vorticity and divergence on the cubed-sphere grid

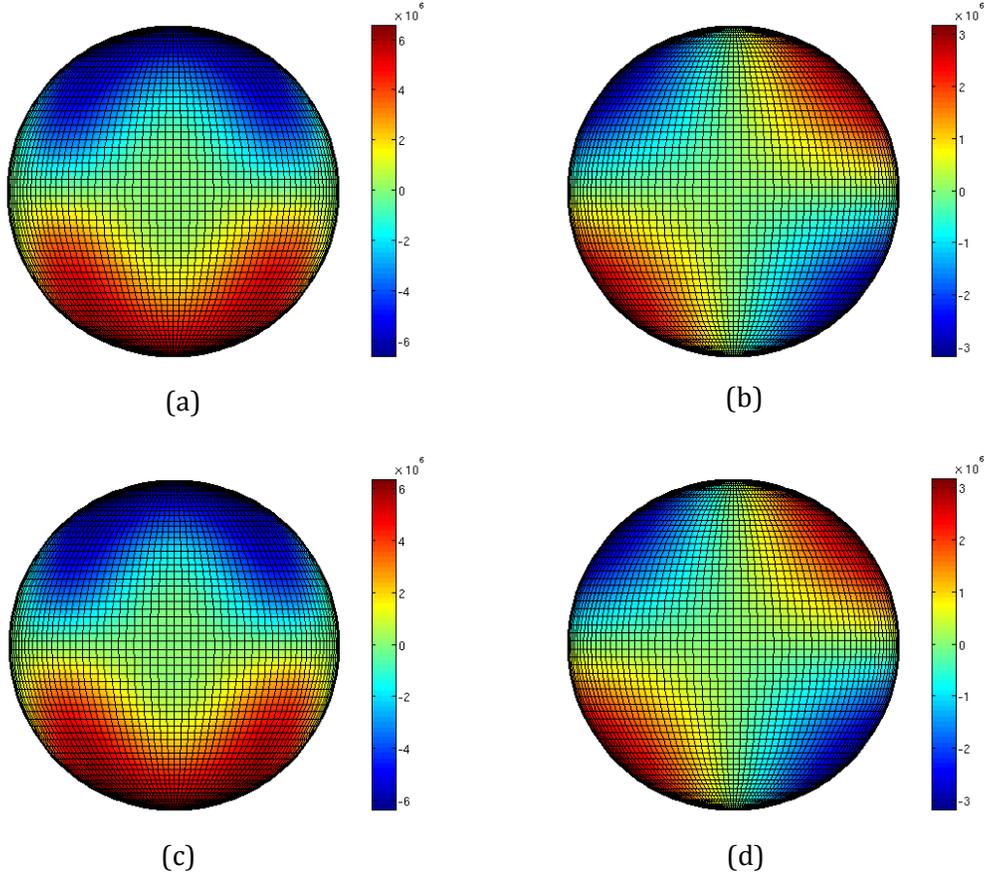


Figure 8: Streamfunction(a) and velocity potential(b) from the ParGAL algorithm and streamfunction(c) and velocity potential(d) from the NCL spherical harmonic algorithm on the T42 CAM-Eul grid for the Rossby wave test case.

Table 4: Errors and convergence rates in the ParGAL computation of streamfunction for the Rossby wave test case.

Mesh	Size	L2 error	Rate	L1 error	Rate
4° FV	48x72	$1.49 \times 10^{-3}$		$1.31 \times 10^{-3}$	
2° FV	96x144	$3.79 \times 10^{-4}$	1.97	$3.33 \times 10^{-4}$	1.98
1° FV	192x288	$9.56 \times 10^{-5}$	1.98	$8.42 \times 10^{-5}$	1.98
T31	48x96	$1.16 \times 10^{-2}$		$7.52 \times 10^{-3}$	
T42	64x128	$7.49 \times 10^{-3}$	1.52	$4.47 \times 10^{-3}$	1.81
T85	128x256	$2.61 \times 10^{-3}$	1.52	$1.26 \times 10^{-3}$	1.82

Table 5: Errors and convergence rates in the ParGAL computation of velocity potential for the Rossby wave test case.

Mesh	Size	L2 error	Rate	L1 error	Rate
4° FV	48x72	$8.63 \times 10^{-4}$		$8.44 \times 10^{-4}$	
2° FV	96x144	$2.09 \times 10^{-4}$	2.06	$2.02 \times 10^{-4}$	2.06
1° FV	192x288	$5.17 \times 10^{-5}$	2.04	$4.97 \times 10^{-5}$	2.04
T31	48x96	$7.82 \times 10^{-4}$		$7.72 \times 10^{-4}$	
T42	64x128	$4.32 \times 10^{-4}$	2.06	$4.26 \times 10^{-4}$	2.07
T85	128x256	$1.06 \times 10^{-4}$	2.04	$1.04 \times 10^{-4}$	2.04

used for the HOMME spectral element model. In the initial implementation of the vorticity and divergence algorithms the nodes of the CAM-SE grid are separated into individual quadrilaterals rather than into high-

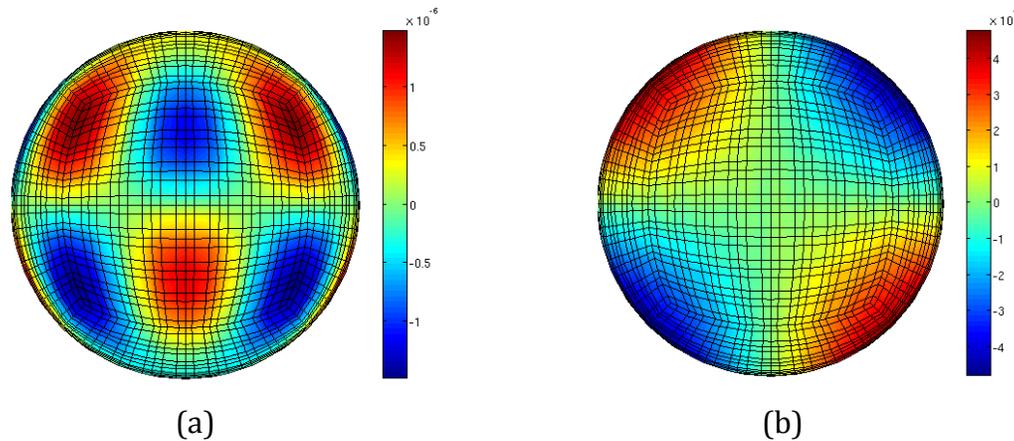


Figure 9: Vorticity (a) and divergence (b) computed on a CAM-SE grid for the Rossby wave test case using ParGAL.

order degrees-of-freedom for the coarse spectral elements. Therefore, bilinear basis functions are used to compute the vorticity and divergence. Future implementations will treat the CAM-SE grid as composed of spectral elements and will use the correct spectral element basis functions to compute these quantities. Plots of vorticity and divergence for the Rossby wave test case and computed for the eight coarse element per face CAM-SE grid ( $np=4, ne=8$ ) are shown in Figure 9. Note that no comparison with NCL is provided since the NCL algorithm is currently restricted to fixed and Gaussian spherical grids.

## ParNCL Development and Release

ParNCL (Parallel NCL) is a parallel version of the NCL interpreter that performs climate data analysis in parallel using ParGAL (and Intrepid) and MOAB.

At the end of our first progress report, we had a prototype version of ParNCL working. We released a beta version (1.0.0b1) to the community on Dec 7<sup>th</sup>, 2012. Several new documents were added and old documents modified in the ParVis wiki related to ParNCL. The wiki now contains a set of NCL scripts that work with both NCL and ParNCL for users to try out. Detailed instructions for building ParNCL were developed for users who want to build their own version.

ParNCL currently supports the same grids as ParGAL, CAM Eulerian Spectral grid, CAM Finite Volume grid and CAM HOMME grid. We will next be adding support for MPAS-ocean and POP grids.

### Current ParNCL features

Other features of our beta version of ParNCL developed over the past year include:

#### Distributed Multidimensional Data Subscripting

ParNCL supports selection of ranges of data from the multidimensional climate data read from a NetCDF file. Users can currently perform the following types of selections on data read from CAM Eulerian Spectral and CAM Finite Volume grids.

- Range subscripting  
Users can provide a beginning and end index of a multidimensional variable followed optionally by a stride value to select a data range.
- Vector subscripting  
Users can explicitly provide the indices of the selection of a multidimensional variable in a vector to get the data slice.

### Analysis Operations

- ParGAL Data Analysis functions  
Users can perform data analysis on data read from a climate data file using NCL functions that have a corresponding ParGAL data analysis function. An example would be computing the average of a variable's given dimension(s) at all other dimensions using the NCL *dim\_avg\_n()* function.
- Simple Math Functions  
Users can perform simple math operations on data using NCL functions that are supported by ParNCL. ParNCL interacts with MOAB and ParGAL to read data from NetCDF files, performs data analysis on the data and stores the result back to the MOAB database. An example of this type of operation would be computing sine of all the values in a multidimensional array using the NCL *sin()* function.
- Simple Math operations  
Apart from using the built-in math functions in NCL, users can perform simple math operations like scaling, addition, subtraction of multidimensional data in ParNCL.
- Viewing and Plotting  
ParNCL supports NCL functions to view data (and metadata) by printing it to stdout. It also supports all NCL graphics functions. The NCL graphics functions execute serially and any distributed multidimensional data passed to these functions is gathered by the interpreter before executing the function.
- New Operations  
ParNCL supports some new operations that were not supported by NCL like calculating vorticity and divergence on a HOMME grid. We have added two new functions *uv2vrA()*, to calculate vorticity from wind components and *uv2dvA()* to calculate divergence from wind components in ParNCL.

The table below gives a summary of the NCL functions currently supported in ParNCL. For more information on these functions please refer to the NCL documentation (<http://www.ncl.ucar.edu/Document/index.shtml>).

NCL function name	Climate Data Grids Supported		
	CAM Eulerian Spectral grid	CAM Finite Volume grid	CAM HOMME grid
addfile	YES	YES	YES
addfiles	YES	YES	YES
dim_avg_n	YES	YES	YES
dim_max_n	YES	YES	YES
dim_min_n	YES	YES	YES
dimsizes	YES	YES	YES
max	YES	YES	YES
min	YES	YES	YES
print	YES	YES	YES
printVarSummary	YES	YES	YES
systemfunc	YES	YES	YES
uv2vrA	YES	YES	YES
uv2dvA	YES	YES	YES
abs	YES	YES	YES
acos	YES	YES	YES
asin	YES	YES	YES
atan	YES	YES	YES
atan2	YES	YES	YES
cos	YES	YES	YES
exp	YES	YES	YES
fabs	YES	YES	YES
floor	YES	YES	YES
log	YES	YES	YES
log10	YES	YES	YES
sin	YES	YES	YES
sinh	YES	YES	YES
ceil	YES	YES	YES
Visualization functions	YES	YES	YES

### ParNCL Software Engineering:

We also modified the NCL test suite to incorporate ParNCL tests. The test suite was modified to run the NCL scripts using ParNCL and verify the results by running the same NCL script using the serial version of NCL.

We also added some new features to the NCL test suite to,

- Allow users to customize the test suite at runtime via a testlist file (Only the tests specified in the testlist file is executed)
- Allow users to specify a query path to the test suite to locate the NCL interpreter

- Specify ParNCL-specific information like number of processors to execute the job, the job launcher etc.

### Near Term Development.

The most immediate development needs for ParNCL are to support more climate data grids and implement data-parallel versions of more NCL analysis functions.

We also plan to add NCL data creation functions like *new()*, *fspan()* to allow users to create distributed data. We also need to provide support for coordinate subscripting (specify coordinates to subscript data) and named subscripting (specify name of dimensions when subscripting data) in future. Support for variable subscripting will also be extended to CAM HOMME grids.

### Data Compression for Ultra-Large data sets

The move toward high-resolution climate models creates bottlenecks for model output and analysis input that cannot be solved by increasing the scale of current parallel file systems. Additionally, the total volume of data generated by the models will quickly exceed our capacity to store the data during simulations or for post analysis. ParVis and the PNNL team are investigating algorithms, based on information theory, that work effectively on floating point climate model outputs. To ease adoption, we are integrating our compression capabilities into the Parallel NetCDF library, which is currently used by existing high-resolution climate codes such as the CESM and GCRM.

Our compression schemes contain two phases: the first phase predicts the next value based on the previous values, the second phase encoded the next value with entropy-based encoding. In the first year, we compared our scheme against other schemes, evaluated various design choices, implemented a prototype and gathered some preliminary results. In the past 15 months, we have made a more detailed analysis of our compression technique.

Since some of our prediction algorithms can predict the exponent part of floating point value and the most significant bit of the significant part of the float point numbers well but not less significant bits, our lossy compression can achieve good compression ratios. Preliminary results, using Matlab and R, show that we can reduce the data by an order of magnitude when the error bound is 10%. Table 6 shows some results for different data using different methods. A more complete experiment and analysis is currently in progress.

	Nearby value + run length encoding	Nearby value + prefix encoding	Nearby value + run length encoding with 10% error bound	Nearby value + prefix encoding with 10% error bound	Gzip
CCSM Temperature	0.513	0.438	0.371	0.126	0.723

<b>CCSM Fractional Cloud Cover</b>	0.437	0.330	0.236	0.082	0.402
<b>CCSM Q Tendency</b>	0.399	0.285	0.218	0.066	0.338
<b>CCSM Cloud Ice</b>	0.317	0.194	0.222	0.041	0.320

**Table 6** Compression ratios for different variables from two different data sets using different compression algorithms. The first three are all lossless and the forth column represents a lossy compression.

## Implementation in PnetCDF

Though compression can be implemented at various points of the I/O stack, in the past year we began implementation within the Parallel NetCDF library. The flexibility of the PNetCDF interface creates a very challenging environment for optimizing compression at the Parallel NetCDF layer. However, we note that climate models typically follow a pattern of appending data, writing without strides, and using collective I/O and will use this information to create a targeted implementation for high-resolution model outputs. In the future, our implementation can be generalized to more usage patterns. Output by PNetCDF is an option in the PIO library used by CESM and used by ParGAL/ParNCL for input and this work will ultimately allow us to keep data compressed throughout the climate model workflow.

## Preliminary Performance Results

For our initial performance tests, we use 28 km data (27 layers) temperature data from the GCRM. We developed a simple Parallel NetCDF benchmark to test reading and writing GCRM data and to examine the performance costs in depth. Our test environment is a linux cluster with fat memory nodes.

Figure 10 (left) shows the read performance. As we can see from the graph, when we only have one MPI process and use the luster parallel file system, the I/O is not the major bottleneck. Decompression causes computation overhead and hence the read compressed version is slower. Note that no optimizations have yet been applied. When we use parallel I/O and have many MPI processes to speed up data retrieval, reading compressed data outperform uncompressed data. For writing, we observe the similar behavior (Figure 10 right).

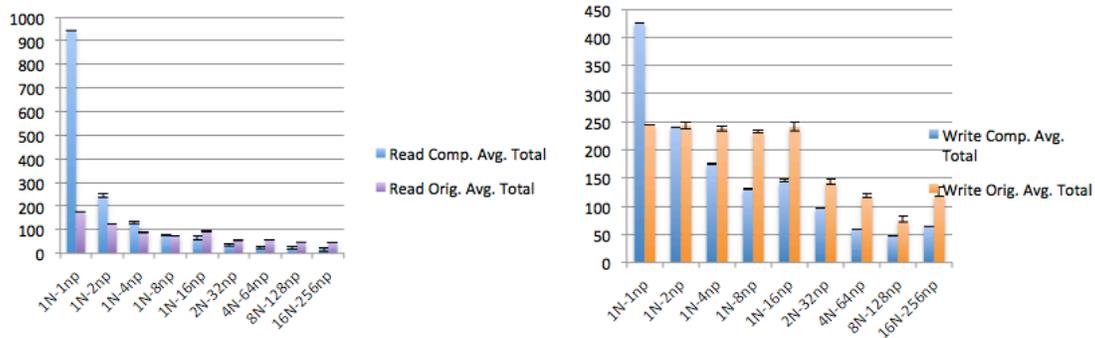


Figure 10: Total read and write performance. The y axis is time in seconds. The x axis shows varying numbers of nodes and processors counts. In all cases, the lustre striping count was set to eight.

Our results to date are promising even without optimization. However, there are many optimizations that need to be evaluated ranging from: choosing an algorithm that best fits a variable and its mesh by examining the header information, simple compiler optimizations, data parallel instruction, restructure of code to reduce branch mispredictions and minimize cache misses and chunking of pipelining.

For correctness testing, we have built the Pagoda parallel analysis library (<https://svn.pnl.gov/gcrm/wiki/Pagoda>) with our compression enabled version of Parallel NetCDF and verified that operators that slice in time, horizontal, and vertical dimensions produce results that when decompressed, exactly match the original data.

### 3D visualization and analysis

ParVis work on advanced visualization techniques for the past 15 months focused on geodesic grids. Geodesic grids are commonly used in climate modeling. Visualization of

**Table 7 The GCRM data sets**

	Low resolution GCRM	High resolution GCRM
Variable file size	31.2MB	5.86GB
Mesh resolution	220km	28km
Time slice	8 days	50 days
Cells	10242	655362
corners	20480	1310720
Edges	30720	1966080
layers	99	61
Variable name	Vorticity   temperature   water_vapor   heat flux   rain water tendency etc.	

large geodesic grid data imposes some unique challenges. First, the data structure of geodesic grids is typically constructed using a recursive refinement procedure on a spherical surface, thus presenting very different geometry properties from other existing unstructured grids. Second, commodity graphics hardware is designed for rendering with trilinear interpolation of planar data. The spherical geodesic

grid data is not organized in such manner, so it cannot be rendered directly by the graphics hardware. Third, even though it is possible to transform geodesic grids into more generally supported grids, such as tetrahedral grids, for visualization, this approach often incurs significant computing and storage overhead, and can become infeasible to process large data from current petascale and future exascale simulations.

The UC Davis team has developed an interactive ray-casting rendering method for visualizing hexagonal grid volume data without first decomposing each hexagonal element into multiple tetrahedral ones. Highly efficient, high quality rendering is achieved with an analytic solution for the interpolation of scalar values with adaptive sampling along a ray within each hexagonal grid cell. A gradient estimation method for rendering hexagonal grid volume data is introduced to achieve smooth shading and highlight to provide important cues for shape and depth. To harnessing the power of GPUs, we organize geodesic grid data structure in GPU memory to best match the original storage format and minimize the data transformation overhead. Table 7 shows the information of the GCRM data set used in our experimental study on a desktop

computer with dual Intel Xeon CPUs, 24GB memory and dual NVIDIA GTX 580 GPUs. The rendering performance results are shown in Table 8.

Table 8: Rendering performance for the low resolution (left) and high resolution GCRM datasets

Variable	CPU	GPU	Variable	CPU	GPU	GPUx2
temperature	29.247s	0.270s	temperature	45.591s	0.783s	0.468s
heat flux	28.177s	0.319s	heat flux	37.199s	0.494s	0.272s
rain water tendency	36.427s	0.285s	rain water tendency	56.247s	0.755s	0.384s
vorticity	24.590s	0.293s	vorticity	59.397s	0.689s	0.460s

The new rendering method can generate high quality visualization of full resolution geodesic grid data, and allows scientists to see greater details from their large climate simulations at interactive rates. Figure 11 compares our method with the conventional tetrahedron based approach and Mean Value Interpolation approach. In addition to superior rendering quality, our method also achieves substantially higher efficiency over the tetrahedron based method and the mean value interpolation method, as shown in Figure 12.

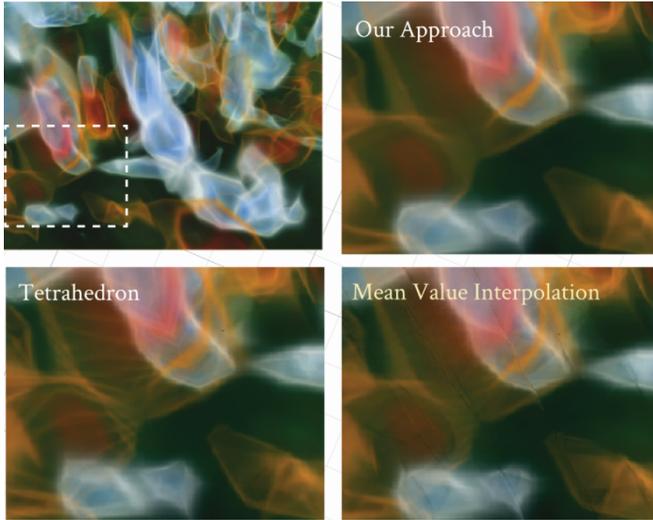
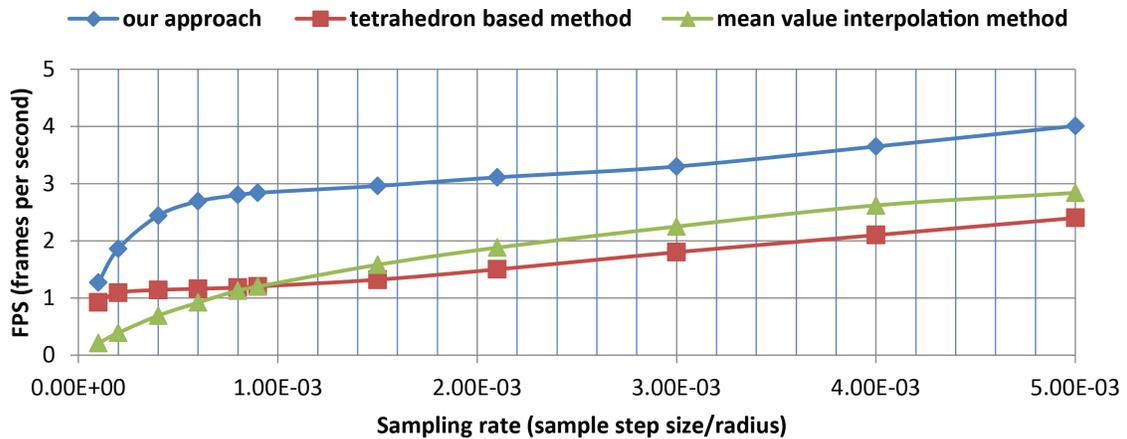


Figure 11: Rendering quality comparison of our approach with the conventional tetrahedron based method and mean value interpolation (MVI) in a close-up views. The images are rendering of high-resolution vorticity data.

interpolation approach.

Figure 12: Performance measures and comparison with the conventional tetrahedron based method and mean value interpolation approach.



Finally, Figure 13 displays mesh rendering, volume rendering of global atmosphere vorticity field with geophysical information superimposed, and a close-up view of a region of interest in the global atmosphere.

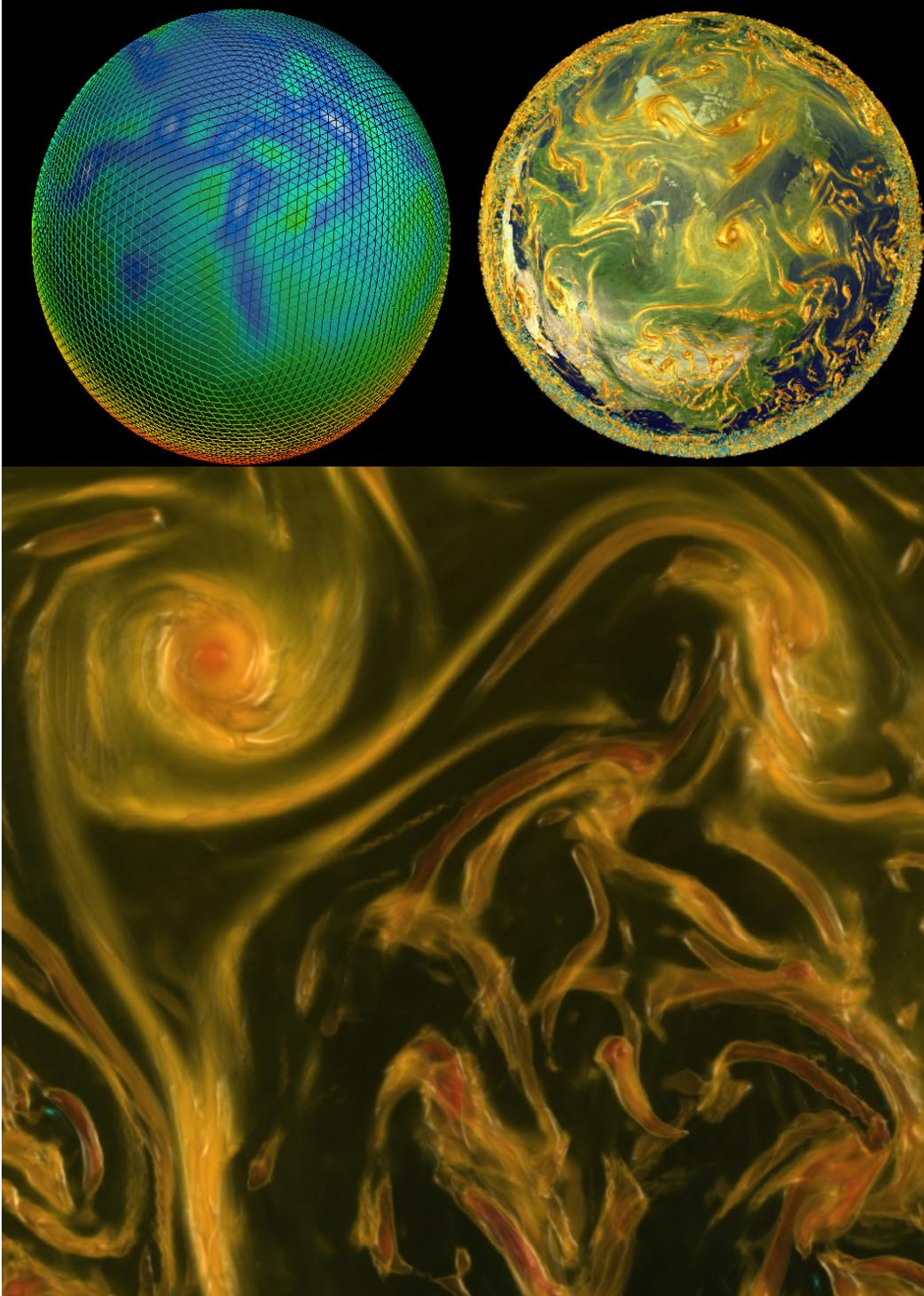


Figure 13: Results from visualization

## Project Management

### Project organization and resources

The PI is responsible for coordinating effort among the various tasks and insuring progress is made on deliverables. The project is spread over 5 institutions and a “lab lead” at each is responsible for coordination of the ParVis members at their respective institutions. The leads are: Robert Jacob (ANL), Pavel Bochev (Sandia), Karen Schuchardt (PNNL), Don Middleton (NCAR) and Kwan-Liu Ma (UC-Davis).

All team members participate in biweekly conference calls devoted to updates and discussion of near-term development. The ANL web and audio service provider, AdobeConnect, is used to facilitate sharing presentations and recording notes from the call. Two mailing lists hosted by Argonne are also used by the team: one for general discussion (parvis) and another for development details and code check-in messages (parvis-dev).

We have biannual all-hands meetings. Our fourth meeting was held March 22-23, 2012, at Argonne National Laboratory and our fifth meeting was October 25-26, 2012, at NCAR.

The PI keeps the ParVis advisory panel (David Randall (CSU) and William Gustafson (PNNL), Gokhan Danabasoglu (NCAR), Cecilia Bitz (University of Washington) and David Lawrence (NCAR)) advised of progress and solicits feedback from them.

The MCS division at Argonne provides resources for software development (svn repository, bug tracking and test/development machines). We have also obtained an allocation of computer time on Argonne’s Fusion cluster for testing on tens to hundreds of processors. ParVis developers have been given access to the Eureka analysis/viz cluster at the Argonne Leadership Computing Facility through the INCITE project led by Warren Washington (time on Eureka is not charged to the project)

### Communication with the broader community

We maintain a website (<http://trac.mcs.anl.gov/projects/parvis>) to both host software we make available for the community and provide notes and material for ParVis team members. Most of the content is world readable except for the repository and the ticket system. ParGAL is open source and instructions are available to download directly from the repository. We also have tarballs available of the ParNCL source and binaries for some systems. We also maintain a one-way mailing list (parvis-ann) that anyone can subscribe to for announcements about ParVis and ParVis software.

The ParVis PI along with the PI’s of the other visualization projects submitted a successful session proposal for the Fall 2011 AGU (Dec, 2011) meeting that informed the community about our efforts. The ParVis project gave a talk at the oral session and presented a poster on the task-parallelism scripts at the poster session. We updated the

CESM community about ParVis with both a poster and a presentation at the 17<sup>th</sup> annual CESM Workshop in June, 2012. We also gave a tutorial on how to run the task-parallel versions of the diagnostic scripts at the workshop that was attended by 35-40 people. The slides from the tutorial are available online. A poster about ParVis was accepted and presented at SC12 in November, 2012. Finally, our AGU session was repeated at the Fall 2012 AGU (Dec, 2012) meeting. Members of the ParVis team presented a talk on Swift and poster on ParVis was also given. The poster presentation included the announcement of the release of the beta version of ParNCL.

To support our users, we have set up a parvis-users mailing list to field questions. We are also maintaining installed versions of ParNCL and the Swift diagnostics on DOE analysis machines (such as lens at ORNL and eureka at ANL).

### **Interaction with other projects**

We have continued to have discussions with the other LAB10-05 projects on how to collaborate.

Members of the BER “Ultra High Resolution Global Climate Simulation” project (PI: Jim Hack, ORNL) have contacted us about using the Swift-based AMWG diagnostics to help analyze their data. We will be working with them to develop diagnostics that are both task parallel and operate directly on the CAM-SE grid. We have examined new diagnostic scripts created by the “Climate Science for a Sustainable Energy Future” project to plot precipitation cycles to see if they can benefit from task parallelism.