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

Argonne National Laboratory: Robert Jacob, Jayesh Krishna, Iulian Grindeanu, Danqing Wu, Tim Tautges, Mike Wilde, Sheri Mickelson
Sandia National Laboratory: Pavel Bochev, Kara Peterson
Pacific Northwest National Laboratory: Jeff Daily, Jian Yin
National Center for Atmospheric Research: Don Middleton, Mary Haley, David Brown, Richard Brownrigg, Wei Huang, Dennis Shea
University of California-Davis: Kwan-Liu Ma, Jinrong Xie

Summary

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

A primary focus of ParVis was introducing parallelism to climate model analysis to greatly reduce the time-to-visualization for ultra-large climate data sets. Work in the first two years was conducted on two tracks with different time horizons: one track to provide immediate help to climate scientists already struggling to apply their analysis to existing large data sets and another 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. In the final 2 years of the project, we focused mostly on the new data-parallel library and associated tools for climate analysis and visualization.

ParGAL

Our library for processing ultra-large climate data on native grids is called the Parallel Gridded Analysis Library (ParGAL) and its development was the main focus of the final 2 years of work.

Refactoring: A careful examination of the existing ParGAL algorithms was conducted to extract generic software behavioral patterns and techniques to make the code more extensible and readable. This work involved introducing new data structures to encapsulate data residing in the mesh and manipulation of the data. The existing algorithms were completely rewritten using the new data structures used for representing data and software design patterns like iterators that help in traversing and manipulating the data. There was no significant performance impact due to the redesign of the algorithms. However, the algorithms now use better software design concepts and are more easily extensible to new meshes (grids).

New functions: two new algorithms, dim_sum_n() and dim_product_n(), were added to the existing list of algorithms supported by ParGAL. The dim_sum_n() algorithm computes the sum of a variable's given dimension(s) at all other dimensions and the dim_product_n() similarly computes the product of a variable's given dimension(s) at all other dimensions.

MOAB/ParGAL development

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

The parallel climate data reader was refactored inside MOAB, to separate different data formats supported, and to allow adding more data formats. Initially, only FV, Euler and HOMME grids were supported. Support for MPAS reader GCRM grids were added conforming to this new design. A GCRM reader, writer, and test suite were successfully added to MOAB. The GCRM grid is now a fully supported addition to the MOAB software.

The initial MPAS reader used trivial partitioning to balance the load on each task. This approach worked fine for meshes with 200K cells, but it could not process in parallel the 65 million element mesh (MPAS 3km glogal grid), due to the very bad locality of data in the file. A new partitioning technique that leveraged the Zoltan recursive coordinate bisection method (RCB), successfully read and processed in parallel the 1.1TB file (See Figure 1).

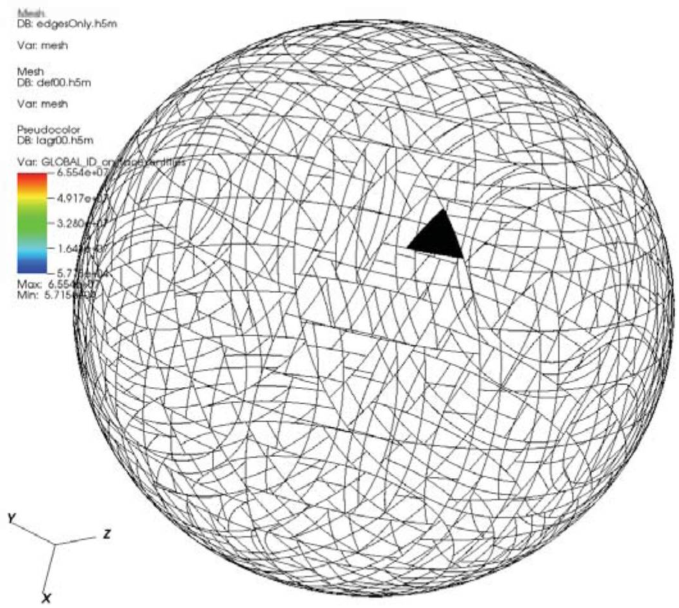

Figure 1: 1024 part partition of the 65M element MPAS mesh. One part is highlighted

A climate data writer was added to MOAB too, which allows saving of the results that can only be calculated with ParGAL so they can be exported to other analysis or visualization tools. The writer follows the new software design of the reader, to allow for easily adding new data formats. Currently, all NetCDF (NC) readers have corresponding NC writers (Euler, FV, Homme, MPAS, GCRM). For unstructured mesh formats, just the variables are written to the new files, the mesh will not be duplicated for the new file, so these new files output from MOAB must be used in conjunction with the original files, that do contain all mesh information.

Additional changes to MOAB include the ability to append new variables to an existing file and support for processing a time series split up over several files.

Intrepid/ParGAL algorithm development

During this reporting period we continued the development, testing, and implementation of parallel algorithms to perform operations currently handled by NCL spherical harmonic functions. Parallel grid-specific algorithms for computing vorticity, divergence, streamfunction, and velocity potential from velocity have been implemented in the ParGAL library for Community Atmosphere Model (CAM) grids. For these grids the finite element method was used to compute L^2 projections and to solve Poisson equations using components of the Trilinos project, including the finite element library, Intrepid, the linear algebra framework, Epetra, and the multi-grid solver, ML. Algorithms corresponding to NCL spherical harmonic functions that have been implemented in the ParGAL library are shown in Table 1. Existing serial functions in NCL are available for CAM finite volume (CAM-FV) and Eulerian (CAM-Eul) grids, but the algorithms for computing these quantities directly on the spectral element (CAM-SE) grid provide new capabilities.

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

NCL Function	Description	Status
uv2dv(F,f,G,g)	Divergence from velocity field	In ParGAL for CAM-FV, CAM-Eul and CAM-SE grids
uv2vr(F,f,G,g)	Vorticity from velocity field	In ParGAL for CAM-FV, CAM-Eul and CAM-SE grids
uv2vr(F,f,G,g)	Vorticity and divergence from velocity field	In ParGAL for CAM-FV, CAM-Eul and CAM-SE grids
uv2sfvp(F,f,G,f)	Streamfunction and velocity potential from velocity field	In ParGAL for CAM-FV, CAM-Eul and CAM-SE grids

Over this reporting period the algorithms for vorticity and divergence were refactored for ease of use and to allow for extensions to algorithms that utilize the finite volume method rather than the finite element method. In addition, the mapping used to convert reference coordinates to physical coordinates on the surface of the sphere for the CAM-SE grid was changed. The previous implementation used a gnomonic cubed sphere mapping and the new method uses an element local mapping. By changing to an element local mapping the method can be used for any unstructured quadrilateral grid on the sphere where velocity values are located at the nodes of the grid.

The algorithms implemented in ParGAL were tested using given velocity fields with known vorticity, divergence, streamfunction and velocity potential and expected convergence under grid refinement was seen. To test parallel performance, scaling runs were completed on Fusion for a 0.1 degree structured CAM-FV grid with 26 vertical levels. The results exhibit good parallel scaling up to 256 processors as shown in Figure

2. For this high resolution grid the parallel algorithm outperforms the serial NCL algorithm even on a single processor, requiring only 712 seconds while execution time for the serial NCL spherical harmonic algorithm is 6568 seconds.

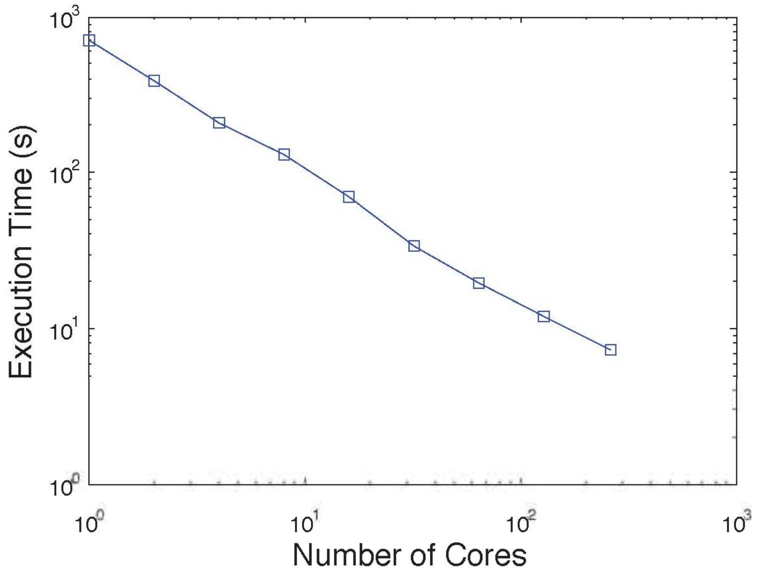

Figure 2. Execution time for computing vorticity from the velocity field for a 0.1degree finite volume grid on the Fusion cluster at ANL.

Parallel Algorithms for MPAS Grids

We also developed a prototype algorithm for computing streamfunction from vorticity for an MPAS Voronoi grid using the finite volume method. The MPAS grid provides vorticity values (ζ) at the nodes of the polygon cells in the mesh (Figure 3). In order to compute the streamfunction (Ψ) we solve a Poisson equation using integration over the dual triangle mesh, shown in red. This algorithm has been implemented in the latest version of ParGAL.

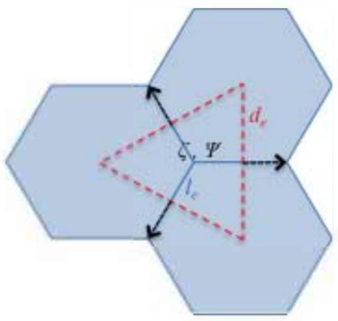

Figure 3. MPAS grid with dual triangle cell shown in red.

The result from a ParGAL streamfunction calculation on the MPAS grid for a Rossby wave test case is shown in Figure 4, along with results from ParGAL streamfunction calculations for CAM-FV and CAM-SE grids for comparison. These plots illustrate the ability of the ParGAL library to perform computations on structured and unstructured grids on the sphere using both finite element and finite volume methods.

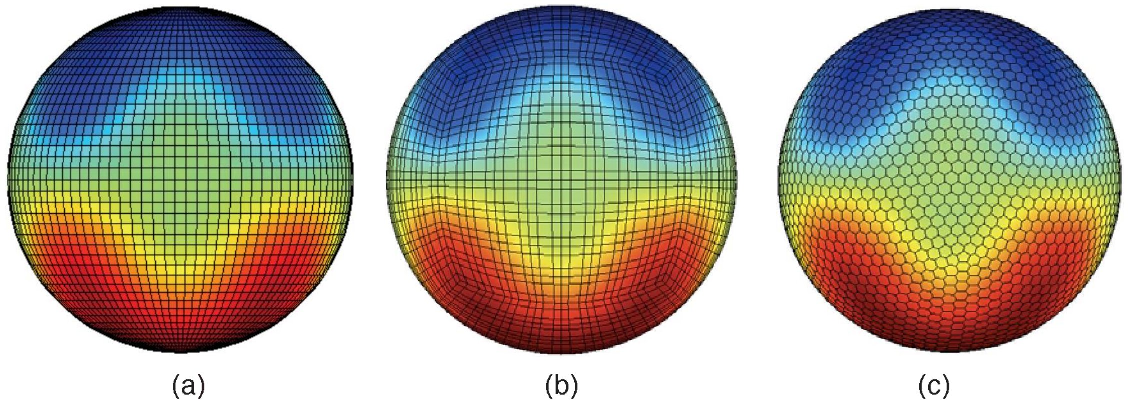

Figure 4. Streamfunction computed using ParGAL algorithms given cell-centered velocities on a CAM-FV grid (a), nodal velocities on a CAM-SE grid (b), and nodal vorticities on an MPAS grid (c).

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. Additional development of ParNCL includes:

MPAS support: All of the ParGAL algorithms have been updated to work with data on MPAS grids. Results are calculated on and saved on the MPAS mesh.

Distributed Multidimensional Data Subscripting for Unstructured Grids: ParNCL now supports data slicing (subscripting) across time and level dimensions for variables read from unstructured grids (HOMME and MPAS). ParNCL continues to support subscripting across all dimensions for distributed data read from structured grids. ParNCL allows users to subscript the distributed multidimensional data read from a NetCDF file by specifying the beginning index, the end index and a stride (Range subscripting) or by explicitly providing the indices of the selection of the data in a vector (Vector subscripting).

Support for variable metadata: The NetCDF variables that represent climate data may have associated ancillary information called attributes that provides more information about the variable. ParNCL now supports variable attributes associated with the variables. The user can also add new attributes to the variables in the user script.

Support for creating new variables on the mesh: ParNCL now allows users to create variables on the mesh using the new() function available in NCL. The data associated with these variables reside on the MOAB mesh and are treated (can be manipulated) as any other data residing on the mesh.

Miscellaneous new algorithms: A parallel version of the NCL gc_inout() algorithm that determines if a list of lat/lon specified points are inside or outside of spherical lat/lon polygon(s) was implemented. The new algorithms dim_sum_n() and dim_product_n() were added into the list of existing algorithms. ParNCL now supports versions of the existing algorithms that retain metadata in the result of a calculation.

Parallel NetCDF writer: ParNCL now uses the NetCDF writer implemented in MOAB to write data in parallel to NetCDF files. A serial NetCDF writer was also implemented to handle variables with data layouts that are not currently supported by the MOAB NetCDF writer.

We made a ParNCL beta release to the community and a 1.0 release is imminent.

Improvements to NCL

In this final period of the NCAR Command Language (NCL) visualization component of the ParVis project, the focus was on refactoring and speeding up the internal algorithms for generating publication-quality raster images of high-resolution structured and unstructured grids generated by the Community Earth System Model (CESM).

The datasets for this project included global hexagonal Model Prediction Across Scales (MPAS) grids generated by the atmospheric and ocean components of the CESM. Grids of two resolutions were included: a lower resolution grid with 2,621,442 cells (approximately 0.16 degrees per cell), and a higher resolution grid with 65,536,002 cells (approximately 0.03 degrees per cell). These grids were chosen because NCL initially took hours to render the lower resolution grid with numerous issues, and was unable to render the higher resolution grid due to the internal triangulation drawing algorithm exceeding its capacity.

The main tasks for this project included:

- Profiling the code to understand the bottlenecks.
- Refactoring the code to work faster in single process mode, using a general scheme that worked for any set of structured or unstructured grids.
- Applying parallelization for further speed-up of the general scheme.
- Ensuring that the refactored code still produced publication-quality visualizations, a key component of NCL.

The bottleneck was caused by a process that 1) projects the cell nodal points into the plane of the map projection, generating a 2D triangulation of the projected points, and 2) creates a connectivity data structure that allows for quick traversal through the mesh.

Profiling this code pointed to the second step as being a major time sink, because it was using a binary tree that became excessively unbalanced.

For the second task, a better key-generation algorithm provided the first big performance improvement, speeding up the rendering of the lower resolution grid from 1.5 hours to just over a minute. In order to draw the higher resolution grid within the limitations of the triangulation code, however, it was necessary to implement a subdivision of the grid into tiles. Initially this was done in data space, but this approach proved to be problematic when transforming into arbitrary map projections. Therefore, this approach was replaced with a tiling of the 2D projected space, a four-step process: 1) points are sorted into their respective tiles, 2) data in each tile are triangulated, 3) connectivity structures are generated, and 4) each tile is plotted to its own section of the in-memory raster image. The complete image was then rendered to the output format (a square PNG image of either 1024 or 6000 pixels).

To ensure a high-quality PNG image, each tile was generated with a bit of overlap with adjoining tiles and the overlapping pixels were thrown away when the tile was inserted into the in-memory image. This eliminates any noticeable edge effects along the boundaries of the tiles.

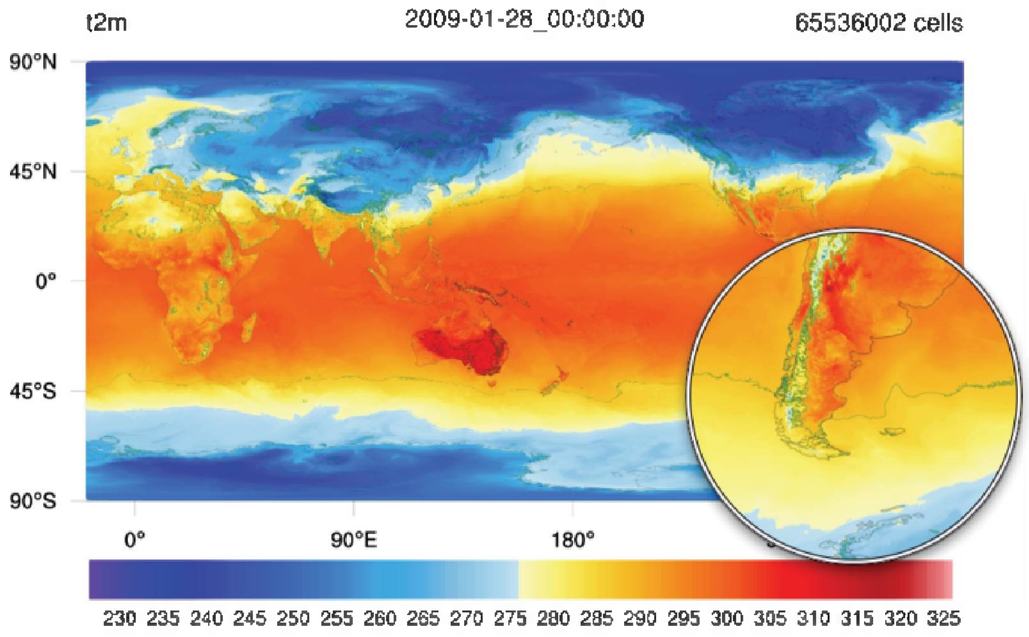

Figure 5. Using the new tiling approach, the first ever NCL graphic of the 65 million MPAS grid was produced, with a rendering time of about 46 minutes in single process mode for a 1024 x 1024 pixel PNG image.

A final improvement to the single process performance came from the realization that, given the correct parameters, the triangulation routine could produce information about the mesh that would allow for the generation of the connectivity data structure directly in linear time. This eliminated the need for the intermediate generation of the binary tree and reduced the rendering time from 46 minutes to just over 6 minutes for a 1024 x 1024 pixel PNG image (Figure 5).

Tiling the image allowed for the introduction of parallelization through OpenMP directives. Some of the lower level Fortran 77 code, which relies heavily on common blocks, is not at all thread-safe, so for the third task, the code had to be rearranged and certain sections marked "critical" in order to make the parallelization work. Given these constraints, the code ran successfully with as many as 36 threads. However, significant additional improvements to the timings seem to end after more than about 16 threads.

By applying OpenMP using 6 threads, the generation time of the 1024x1024 image of the 65 million-cell grid was reduced to about 2.5 minutes. See Table 2 for a summary of the observed speed-ups under different conditions. High-quality images are important to NCL users, so we included timing results for both 1024 x 1024 and 6000 x 6000 pixel images.

NCL Test Case	**PNG Size**	**CPU seconds** *Lower resolution grid*	**CPU seconds** *Higher resolution grid*
Before any improvements	1024	~5400	Not possible
Before any improvements	6000	Not attempted	Not possible
After refactoring, single threaded	1024	23.6	406.3
After refactoring; single threaded	6000	43.4	430.3
After refactoring; OpenMP applied with 6 threads	1024	15.1	175.8
After refactoring; OpenMP applied with 6 threads	6000	23.3	233.6
After refactoring; OpenMP applied with 16 threads	1024	12.0	146.8
After refactoring; OpenMP applied with 16 threads	6000	21.9	173.6

Table 2: A summary of the NCL speed-up rates observed when executing an NCL script to create color-filled PNG images of the lower and higher resolution MPAS grids.

For the final task, the quality of these images was improved by refactoring the contour line and label generating code to work with the tiling mechanism. These features have not been modified to work with OpenMP, but they will draw correctly in the context of splitting the mapped area into tiles.

Further improvements are possible with more refactoring of the code, including rewriting more of the Fortran 77 code with thread safety in mind, and parallelizing the cairo graphics library.

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 **compression** 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.

In our last year, we mainly focused on producing publications to disseminate our research results. We published two full research papers and one poster paper, respectively

- T. Bicer, J. Yin and G. Agrawal, "*Improving I/O Throughput of Scientific Applications using Transparent Parallel Compression*", in Proceedings of the International Symposium on Cluster, Cloud and Grid Computing (CCGrid'14), Chicago IL, May 2014 (283 submitted, 54 accepted, 19.1% acceptance rate),
- Dongfang Zhao, Jian Yin, Kan Qiao, Ioan Raicu. "*Virtual Chunks: On Supporting Random Accesses to Scientific Data in Compressible Storage Systems*", IEEE International Conference on Big Data 2014; (18% acceptance rate), and
- Dongfang Zhao, Jian Yin, Ioan Raicu. "*Improving the I/O Throughput for Data-Intensive Scientific Applications with Efficient Compression Mechanisms*", Poster at SC 2013.

Our two full research papers were accepted into conferences with acceptance rates less than 20% and our CCGrid paper was nominated for best paper.
In our CCGrid paper, we describe how compression can improve end-to-end performance of scientific applications. We present an integrated architecture where compression is transparently implemented at the application library level. In this paper, we present our results for both reads and writes. Parallel writes are particular challenging with compression. We organized and analyzed the techniques that we developed in the previous year and experimental results and present the performance tradeoff between dense storage and sparse storage.

In our SC poster and IEEE BigData paper, we present an architecture that integrates compression at the file system level. Our SC poster gives a brief overview while our IEEE BigData paper gives more details. We also submitted a journal paper to IEEE Transactions on Services Computing and it is currently in review. In these papers, we describe a concept called virtual chunks aiming to support efficient random accesses to the compressed scientific data without sacrificing its compression ratio. Implementing compression at the file system level is particularly challenging. Compressing the entire data can result in a good compression ratio, however any retrieval request triggers the decompression from the beginning of the compressed file. On the other hand, block-level compression provides flexible random accesses to the compressed data, but introduces extra overhead when applying the compressor to each block that results in a degraded overall compression ratio. In either case, the storage systems result in retrieving more blocks than are needed and hence the end-to-end application performance degrades. Our proposed virtual chunk architecture is designed to address this problem. In the virtual chunk architecture, we can start decompression from multiple starting points within the data while still allowing data to be stored continuously in storage devices. This enables random access of compressed data and avoids the pitfall of having to retrieve more storage blocks than are needed. Since the number of storage blocks that need to be retrieved determines the I/O overhead, we significantly reduce I/O overhead and improve application throughput. Another advantage of implementing parallel compression at the file system level is that it is transparent; no application or application library changes are needed and hence can save much time for application developers and make deployment easier.

Pagoda command-line tools

ParGAL excels at complex spatial operations in parallel on ultra-large climate data because of its detailed description of the mesh with MOAB. But simple time averages can, in many cases, use tools without detailed mesh descriptions. The main impediment to using the Pagoda parallel command-line tools for this purpose was its dependence on the Global Arrays library for parallelization. Efforts were made to eliminate this dependency in a future release of the Pagoda tools by focusing efforts on an MPI-only runtime for the Global Arrays library. The next release of Pagoda will only depend on the Parallel NetCDF library and MPI which will significantly reduce its barriers to adoption.

In addition, the work on Pagoda as well as our compression work was presented at the American Geophysical Union 2013 Fall Meeting. Our emphasis on reducing I/O costs using parallel I/O was well received by attendees. As a result, parallel I/O will be a point of emphasis in ongoing NetCDF Operators (NCO) development by Zender et al. done under the ACME project. This may result in the diminished use of our Pagoda tools, however our influence in utilizing parallel I/O in the more widely adopted tools used by the climate community cannot be overlooked.

Additional approaches for reducing I/O in climate models

Further efforts to support the GCRM grid and high performance I/O were realized with a journal submission to the International Journal of High Performance Computing Applications.

- Lynn Wood, Jeff Daily, Michael Henry, Bruce Palmer, Karen Schuchardt, Donald Dazlich, Ross Heikes, and David Randall. "A global climate model agent for high spatial and temporal resolution data," International Journal of High Performance Computing Applications 1094342013518808, first published on January 17, 2014 doi:10.1177/1094342013518808.

In this paper we address the issue that fine cell granularity in modern climate models can produce terabytes of data in each snapshot, causing significant I/O overhead. Our approach uses a method of reducing the I/O latency of high-resolution climate models by identifying and selectively outputting regions of interest. Using the GCRM model and running with up to 10,240 processors on a Cray XE6, this method provides significant I/O bandwidth reduction depending on the frequency of writes and the size of the region of interest. The implementation challenges of determining global parameters in a strictly core-localized model and properly formatting output files that only contain subsections of the global grid are addressed, as well as the overall bandwidth impact and benefits of the method. The gains in I/O throughput provided by this method allow dual output rates for high-resolution climate models: a low-frequency global snapshot as well as a high-frequency regional snapshot when events of particular interest occur.

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), Jeff Daily (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. Regular telecons continued through May, 2014. 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).

Our last all-hands meeting was in October, 2012.

The PI keeps the ParVis advisory panel (David Randall (CSU), 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 ParGAL source will be moved to github after the 1.0 release.

In addition to the papers mentioned above, the ParVis submitted a successful session proposal for the Fall 2013 AGU (Dec, 2013) meeting that informed the community about new approaches to big data analysis in climate. The ParVis project gave a talk on data-parallel algorithms at the oral session and presented an overview poster at the poster session. We updated the CESM community about ParVis with posters and presentations at the 17th and 18th annual CESM Workshops in June, 2013 and 2014.

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.