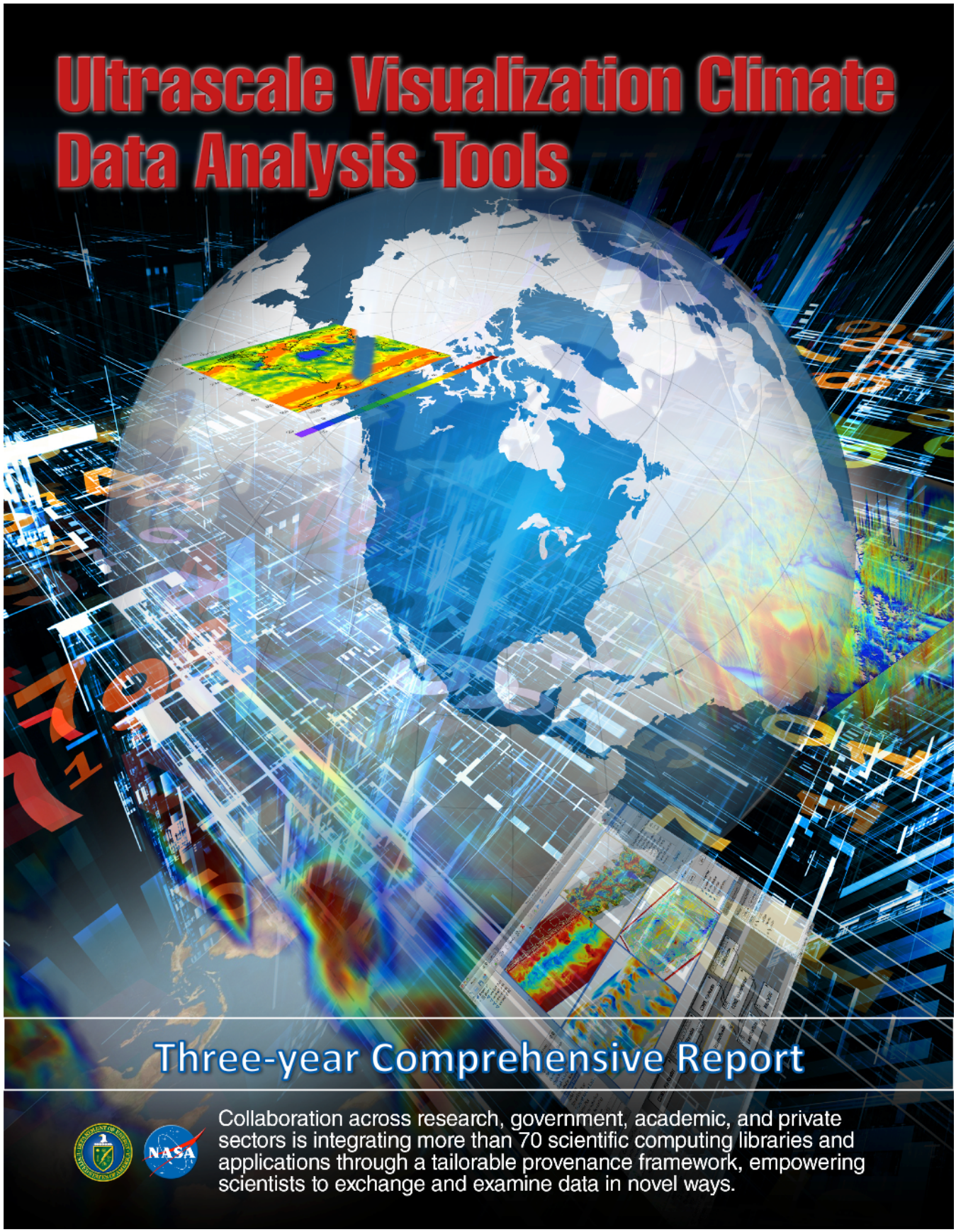
# Ultrascale Visualization Climate Data Analysis Tools



## Three-year Comprehensive Report

Collaboration across research, government, academic, and private sectors is integrating more than 70 scientific computing libraries and applications through a tailorable provenance framework, empowering scientists to exchange and examine data in novel ways.

LLNL-TR-643624

**Cover:**
UV-CDAT offers the Department of Energy's (DOE) Office of Biological and Environmental Research, the National Aeronautics and Space Administration (NASA), the National Oceanic and Atmospheric Administration (NOAA), and other institutions extreme value analysis plotting using an array of visualization tools, such as CDAT, DV3D, VisIt, and ParaView. The customizable provenance and workflow framework empowers scientists around the world to exchange analysis and examine data in novel ways.

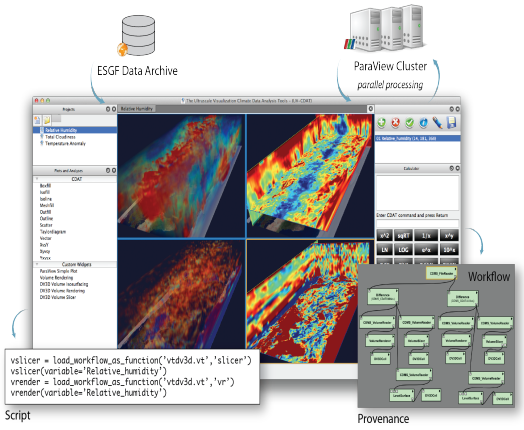
**Image credits:**
This work is supported by the Director, Office of Science, Office of Biological and Environmental Research, of the US Department of Energy, under contracts DE-AC02-05CH11231 and DE-AC52-07NA27344 and by the National Aeronautics and Space Administration. This research used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory, which is supported by the DOE Office of Science under Contract No. DE-AC05-00OR22725.

# Department of Energy's Biological and Environmental Research Ultra-scale Visualization Climate Data Analysis Tools (UV-CDAT)

***Three-year Comprehensive Report***
***October 1, 2010, through September 30, 2013***

***Principal Investigator***
Dean N. Williams[3]

***The UV-CDAT Team***
Andrew Bauer[1], Aashish Chaudhary[1], Berk Geveci[1],
Harinarayan Krishnan[2],
David Bader[3], Timo Bremer[3], Charles Doutriaux[3], Daniel Fedor-Thurman[3], Matthew Harris[3],
Elo Leung[3], Renata McCoy[3],
James Ahrens[4], Curt Canada[4], Phil Jones[4], Boonthanome Nouanesengsy[4], John Patchett[4], Sean Williams[4],
Thomas Maxwell[5], Gerald Potter[5],
Cecelia DeLuca[6], Ryan O'Kuinghttons[6], Robert Oehmke[6],
David Pugmire[7], Galen Shipman[7], Brian Smith[7], Chad Steed[7],
Ben Burnett[8], Aritra Dasgupta[8], Tommy Ellqvist[8], David Koop[8], Emanuele Marques[8], Jorge Poco[8], Rémi Rampin[8], Claudio Silva[8], Huy Vo[8],
David Kindig[9], Alexander Pletzer[9],
Cameron Christensen[10], Sidharth Kumar[10], Valerio Pascucci[10], Giorgio Scorzelli[10], Brian Summa[10]




**Building the future of interactive analysis and visualization one component at a time**

[1] Kitware, Inc
[2] Lawrence Berkeley National Laboratory (LBNL)
[3] Lawrence Livermore National Laboratory (LLNL)
[4] Los Alamos National Laboratory (LANL)
[5] National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC)
[6] National Oceanic and Atmospheric Administration (NOAA)
[7] Oak Ridge National Laboratory (ORNL)
[8] Polytechnic Institute of New York University (NYU-Poly)
[9] Tech-X Corporation
[10] University of Utah

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

# Contents

# 1 Abstract

For the past three years, a large analysis and visualization effort—funded by the Department of Energy's Office of Biological and Environmental Research (BER), the National Aeronautics and Space Administration (NASA), and the National Oceanic and Atmospheric Administration (NOAA)—has brought together a wide variety of industry-standard scientific computing libraries and applications to create Ultra-scale Visualization Climate Data Analysis Tools (UV-CDAT) to serve the global climate simulation and observational research communities. To support interactive analysis and visualization, all components connect through a provenance application programming interface to capture meaningful history and workflow. Components can be loosely coupled into the framework for fast integration or tightly coupled for greater system functionality and communication with other components. The overarching goal of UV-CDAT is to provide a new paradigm for access to and analysis of massive, distributed scientific data collections by leveraging data architectures located throughout the world. The UV-CDAT framework addresses challenges in analysis and visualization and incorporates new opportunities, including parallelism for better efficiency, higher speed, and more accurate scientific inferences. Today, it provides more than 600 users access to more analysis and visualization products than any other single source.

*The entire UV-CDAT team would like to sincerely thank our funding sponsors in the DOE BER Office—as well as our national and international collaborators, stakeholders, and partners—for allowing us to work with you and the climate science community these past few years.*

# 2 Executive Summary

The mission of Ultra-scale Visualization Climate Data Analysis Tools (UV-CDAT) is to provide access to large-scale data analysis and visualization tools for the climate modeling and observational communities. UV-CDAT's specific goals were to develop the software and workflow applications needed to:

- Integrate U.S. Department of Energy's (DOE's) climate modeling and measurements archives.
- Develop infrastructure for national and international model–data comparisons.
- Deploy visualization, diagnostic, and analysis tools with easy to use interfaces for very large, high-resolution climate data sets that meet the growing demands of the data-driven climate science community.

The integrated, cross-institutional UV-CDAT consortium of computational and science teams was funded under the DOE's Office of Biological and Environmental Research (BER), in direct support of its climate science mission. The team's members included four DOE national laboratories (Lawrence Berkeley [LBNL], Lawrence Livermore [LLNL], Los Alamos [LANL], and Oak Ridge [ORNL]); two universities (the Polytechnic Institute of New York University [NYU-Poly] and the University of Utah; the private company Kitware, Inc.; the private company Tech-X Corporation funded by the National Oceanic and Atmospheric Administration (NOAA); and the Goddard Space Flight Center (GSFC) funded by the National Aeronautics and Space Administration (NASA).

Led by LLNL, the UV-CDAT team developed and delivered a production environment for data exploration and analysis of large-scale geospatial data. The environment has come to include such national and international analysis and visualization products as the Climate Data Analysis Tools (CDAT), VisTrails, ParaView, Data Visualization 3D (DV3D), VisIt, the Earth System Modeling Framework (ESMF), and R. The resulting integrated enterprise is a powerful and complete front-end of a mixed set of visual-data exploration and analysis tools.

Working directly with climate science analysis projects, the consortium deployed computational resources essential to a range of stakeholders, including scientists, policymakers, and the general public. Members of this consortium collaborated with other institutions and universities in researching data discovery, management, visualization, workflow analysis and provenance.

We worked closely with scientific programs funded by DOE's BER to advance the development of state-of-the-art tools in support of BER's science requirements, which included making publicized data archives (such as model intercomparison project data, observational data, and very-high-resolution climate model simulations) more useful to stakeholders (e.g., climate researchers). In addition, we met specific needs of national and international climate projects by integrating and developing tools and techniques suitable for large data sets in a familiar, distributed, federated infrastructure. Finally, we provided international climate centers and U.S. government agencies with a collection of climate data analysis tools and diagnostic methods for ultra-large climate data sets—in particular, sequential and parallelized tools and methods needed to support working groups focused on model development and diagnosis.

The team continues to leverage existing Kitware support tools used for UV-CDAT development, such as CMake, CTest, and CDash. In addition, the UV-CDAT team focused on community outreach, by holding workshops and online classes, publishing articles in leading journals and chapters in books, and developing a governance document to facilitate community goodwill and collaboration.

We are submitting this comprehensive report at the end of our original project's funding cycle. At the same time, we are seeking to integrate our work into the DOE-Earth System Modeling (ESM) project, which would allow us to maintain and enhance UV-CDAT production and operation and continue to serve the climate science community.

# 3 Overview

## 3.1 Problem Statement

As stated in our original proposal, climate science has moved beyond the physical climate to include detailed carbon cycle hydrology and biogeochemical and other feedback processes within new Earth system models. As a result, a large number of new variables are required to track tracers in current and future Earth system models. In addition, observational programs and field campaigns are generating increasingly rich and more complex data sets. Advanced tools for managing, analyzing, and visualizing ultra-scale climate data are required to maintain the rapid progress in our scientific understanding and prediction of climate change and its impacts across all scales and to apply this understanding to the decision-making process.

With that, the Ultra-scale Visualization Climate Data Analysis Tools (UV-CDAT) was established to address the needs of modern-day climate data centers and climate researchers. Specifically, UV-CDAT addresses the needs of both data centers and researchers in the area of analysis and visualization of large and complex data sets. Under the leadership of the DOE BER, the UV-CDAT team consisted of four DOE national laboratories (Lawrence Berkeley [LBNL], Lawrence Livermore [LLNL], Los Alamos [LANL], and Oak Ridge [ORNL]); two universities (Polytechnic Institute of New York University [NYU-Poly] and the University of Utah; two private companies (Kitware, Inc. and Tech-X Corporation), National Oceanic and Atmospheric Administration (NOAA); and the National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC). Although this development effort was primarily coordinated nationally, it is beginning to attract a broader base of international developers interested in extending UV-CDAT for their climate modeling efforts, including the British Atmospheric Data Centre and the University of Edinburgh School of Geosciences Grant Institute.

The work on UV-CDAT has resulted in new ways of looking at data, empowering scientists to pull together disparate tools for unique climate science undertakings. Through this UV-CDAT effort, the team was able to deliver on the following proposal goals:

- An overall architecture for incorporating existing and future software components.
- Analysis and visualization of heterogeneous data sources (simulations, observations, and re-analysis)—data analysis that cut across multiple disciplinary domains.
- Workflow and provenance for user reproducibility.
- Parallelism and “big data” distributed analytics.
- New frontiers for diagnostics (such as ensemble analysis, uncertainty quantification, and metrics) computation.
- On-going support for climate/atmospheric model and data evaluation activities and projects, such as the Intergovernmental Panel on Climate Change (IPCC) assessment report, DOE’s Earth System Modeling project, NASA Center for Climate Simulation (NCCS), and NASA Global Modeling and Assimilation Office.

As a result of the substantial analysis and visualization requirements for the ESM community, particularly at global and continental scales, we built and delivered an advanced application—UV-CDAT—that can locally and remotely access ultra-scale globally federated data archives, provide high-performance parallel analysis and visualization capabilities to the desktop of a climate scientist, and integrate community-proven tools to help scientists make informed decisions on climate change consequences. UV-CDAT’s sophisticated software integrates over 70 distinct packages and libraries, includes over a dozen national participating developing sites and a growing number of users (over 600), and speeds up analysis and visualization routines. The UV-CDAT software stands out as an emerging collective of analysis and visualization products to lead into the future.

UV-CDAT is built on the integration of software and hardware resources spread across DOE, NASA, and university data centers carrying out climate research. It builds on the following key software technologies:

- CDAT: developed at LLNL for the analysis, visualization, and management of large-scale distributed climate data.
- ParaView: an open-source, multi-platform, parallel-capable visualization tool with recently added specialized capabilities to better support specific needs of the climate science community.
- VisTrails: an open-source scientific workflow and provenance management system to support data exploration and visualization.
- VisIt: an open-source, parallel-capable, visual data exploration and analysis tool capable of running on platforms ranging from laptops to DOE's largest supercomputers.

These combined tools, along with others (e.g., R), and custom packages (e.g., NASA's DV3D and NOAA's ESMF), form UV-CDAT and provide a synergistic approach to the scientific analysis and visualization of climate model and observational data.

The team members worked across institutional boundaries to develop and integrate software packages under one framework that facilitate climate research and enable scientists to use them with little or no effort. Close group collaborations and marathon national face-to-face meetings, teleconferences, and coding and debugging sessions were held often to meet the requirements for this ambitious endeavor. Group software development and milestones include the following:

- Interactive and batch analysis, diagnostics, and visualization development, including support for exploring new ways of interacting with comparative visualization and statistical analyses.
- Robust tools for regridding, reprojection, and aggregation, including support for unstructured grids and non-gridded climate data, and geospatial formats often used for observational data sets.
- Graphical user interface (GUI) designed for ease-of-use.
- Workflow analysis and provenance management.
- Parallel visualization and analysis tools (exploiting parallel input/output [I/O]).
- Monitoring and metrics, for reporting data and system use.
- Website and mailing lists, for helping users address questions and issues pertaining to the UV-CDAT system.

UV-CDAT's success can be measured by its expanding use. It is now integrated with the Earth System Grid Federation (ESGF) as a front-end access mechanism to acquire data for analysis and visualization and as a prototype back-end tool used to reduce data sets and return visualization products. It is also expanding its use to other DOE- and NASA-funded projects as the cornerstone of interagency proposed projects. In particular, two future projects under DOE's ESM efforts aim to use UV-CDAT to deliver new capabilities that will further facilitate interactive and visual exploration and diagnostics of simulation and observational output. These projects have a joint vision for large-scale visualization and analysis of climate data and will work to organize and expand UV-CDAT's capabilities.

## 3.2 Use Cases

The UV-CDAT project was founded on a large set of use cases, which have significantly driven its development activities and prompted it to support both interactive and batch processing. For a climate scientist to be able to use the tool, interactive processing requires performance sufficient to keep the scientist engaged. We envisioned batch processing to involve submitting a job to a supercomputer much as a simulation is submitted. For interactive use cases, the team assembled a front-end GUI for any number of possible scenarios and combinations thereof. For

batch processing, the time constraints for the interactive use case are obviated, but the requirement for scripting workflows becomes paramount.

The typical UV-CDAT large-scale batch use case is one involving a large number of time steps that process a small spatial granularity. Independent temporal parallelism is embarrassingly parallel with respect to time, i.e., an individual time step is not dependent on any other time step to produce an analysis product such as a time-evolution video of sea surface height. Time-dependent use cases require other time steps for the production of such analysis products as a time averaging operation or ensemble analysis.

This distinction becomes important because it affects the chosen parallel architecture while maintaining performance. UV-CDAT performs numerous parallel batch tasks that can generally be divided into the set of parallel use cases, some shown in Figure 1.

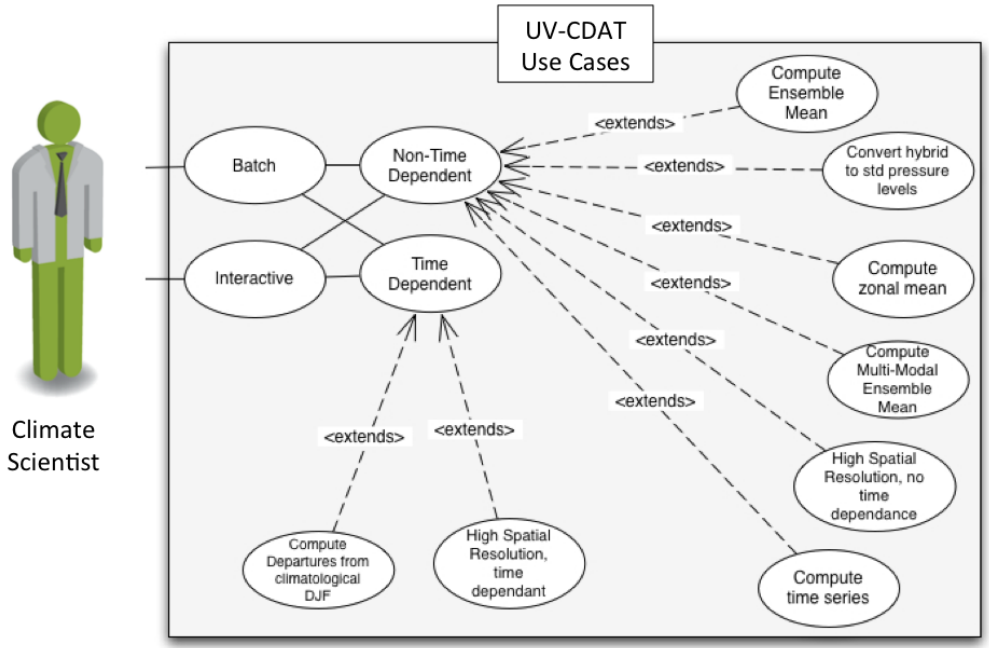



***Figure 1***. *Use case diagram for UV-CDAT.*

These use cases are for tasks commonly performed by climate model analysts that require explicit spatial, temporal, spatio-temporal, and multi-attribute considerations. Most of these use cases have at least one embarrassingly parallel dimension. These are time (3, 4, 6, 8), space (3, 4, 5, 6, 7), ensemble member (2, 3, 5, 6, 7, 8), or model (1, 2, 3,5, 6, 7, 8). Although many of these operations have embarrassingly parallel aspects, climate data in parallel often create a data-intensive problem with I/O scaling difficulties. Many of the parallel architectures developed for UV-CDAT account for this problem. The use cases that have driven UV-CDAT's parallel development are:

- Use case 1: High-resolution, parallel, image sequence.
- Use case 2: High-resolution, parallel, time average.
- Use case 3: Compute ensemble mean.
- Use case 4: Compute multi-model ensemble mean.
- Use case 5: Compute departures from a climatological boreal winter (DJF).
- Use case 6: Convert from hybrid to standard pressure levels.
- Use case 7: Compute a time series of a regional average.

- Use case 8: Computing a zonal mean.
- Use case 9: Batch processing.
- Use case 10: Interactive processing.
- Use case 11: Time-dependent processing.
- Use case 12: Time-independent processing.

For more detailed information on use cases, see Appendix C and also visit the UV-CDAT use case website: http://uv-cdat.org/wiki/UseCases.

## 3.3 Project Success and Accomplishments

The ultimate goal of this project was to address the particular requirements and ultra-scale comparative capabilities needed for access, analytics, and visualization of climate data sets for BER and NASA projects. Many milestones have been archived and, if funding persists, many more will be on the way. The UV-CDAT team delivered analysis and visualization products associated with these milestones, listed under section 3.1, to the research community. UV-CDAT was developed with the following user groups in mind:

- Scientists (working group I types).
- Impacts, adaptations (working group II types).
- Mitigation (working group III types).

BER efforts targeted the scientist user group only. The working groups focused on climate impacts and mitigation were primarily targeted by other funding agencies such as NASA, NOAA, universities, and private companies.

Other DOE projects (e.g., the multiple BER climate modeling project initiatives and the ESGF) have come to rely on UV-CDAT to provide visualization, analysis, and diagnostics for the climate science community. As scientists increasingly depend on more sophisticated, next-generation atmospheric–ocean–sea–ice–land models, they will have a greater need for analysis and visualization resources. Central to the current and future dissemination of data is ESGF, and central to serving up analysis and visualization products to ESGF is UV-CDAT. Although ESGF focuses primarily on data management, it allows analysis tools such as UV-CDAT to access its data holdings and produce server-side products.

### *3.3.1 Balanced Research and Development: Overall Architecture Design*

UV-CDAT is a workflow-based, provenance-enabled system that integrates climate data analysis libraries and visualization tools in an end-to-end application. UV-CDAT lets users build complex data analysis and visualization workflows that use predefined components for data transformations, collect data from disparate data sources outside ESGF, and allow for user-defined local and remote processing steps.

The design of UV-CDAT, shown in Figure 2, incorporates the following requirements:

- Interactive and batch operations.
- Workflow analysis and provenance management.
- Parallel visualization and analysis tools (exploiting parallel I/O).
- Local and remote visualization and data access.
- Comparative visualization and statistical analyses.
- Robust tools for regridding, reprojection, and aggregation.

- Support for unstructured grids and non-gridded observational data, including geospatial formats often used for observational data sets.

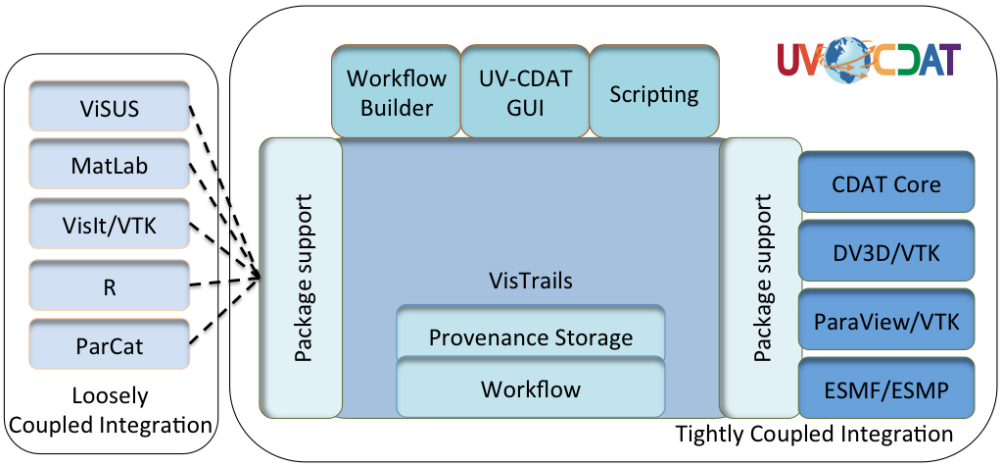



***Figure 2.*** *The UV-CDAT architecture can integrate framework components, either tightly or loosely coupled. From this design, other packages can be joined seamlessly.*

To achieve this design, UV-CDAT uses several components, including CDAT, VisTrails, DV3D, ParaView, and VTK, to provide high-performance, parallel-streaming data analysis and visualization of massive climate data sets (see Figure 2). These core components are tightly coupled for greater system performance.

In addition, UV-CDAT is a very flexible and extensible system. Using VisTrails's package mechanism, it is simple to loosely integrate (or incorporate) other tools, including VisIt, ViSUS, R, and MatLab for data analysis and visualization without modifying the system core. This package mechanism enables developers to expose their libraries (written in any language) to the system by a thin Python interface through a set of VisTrails modules. Users are able to interact with the system in different ways, by using the UV-CDAT GUI, the VisTrails' workflow builder, or through Python scripts.

We believe that providing these different ways of integration and interaction, including a simple but powerful GUI, reduces the barriers for adoption. In addition, relatively simple interfaces are needed for other target audiences (for example, adaptation and mitigation researchers and decision makers).

### *3.3.2 Success of Software Integration*

Nationally, the UV-CDAT software development team (and indeed the community) has accomplished something that has never before been attempted, much less completed at this level of software engineering for the climate community: the integration of more than 70 disparate software packages and libraries for large-scale data analysis and visualization, as depicted in Figure 3. Table 1 includes a list of the 70-plus software packages incorporated into UV-CDAT. Reasoning behind the integration of 70-plus packages is to give the end-user the flexibility to pick and choose the best of each package or to utilize a feature in one package that is lacking or missing in another. The downside to this approach is that each primary package (shown in Figure 2) has a substantial list of sub-packages and libraries. In some cases, users will not want to install every primary package. Thus, this installation use case necessitated the need for the "UV-CDAT-Lite" build. "UV-CDAT-Lite" enables more flexibility at configuration and allows the installer to choose only the primary packages they desire (e.g., CDAT, DV3D, ParaView, R, etc.). By choosing subset of primary packages, a smaller set of sub-packages and libraries will be built and installed as well. This installation procedure has the extra benefit of increasing robustness by enabling less mature packages to be disabled when not needed.

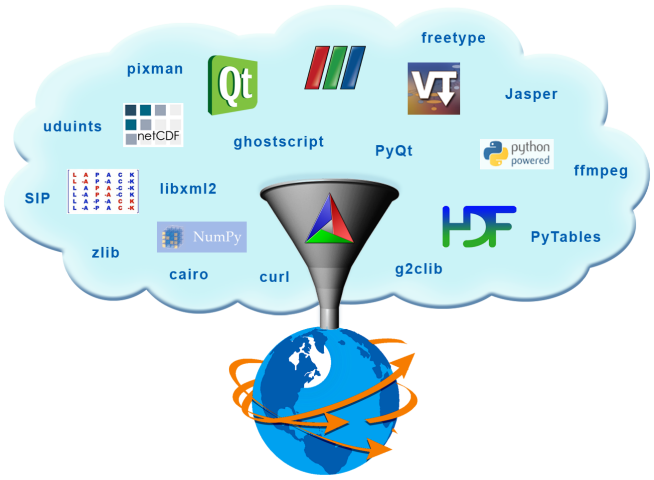



***Figure 3.*** *Image depicts the wide variety of industry-standard scientific computing libraries and packages funneling down to create UV-CDAT to server the global climate simulation and observational research communities.*

***Table 1.*** *UV-CDAT builds upon over 70 external software packages.*

| | | | | |
|---|---|---|---|---|
| basemap 1.0.5 | pkgconfig 0.23.0 | termcap 1.3.1 | readline 6.2 | libxml2 2.7.8 |
| libxslt 1.1.26 | CURL 7.22.0 | YASM 1.2.0 | ffmpeg 0.11.1 | zlib 1.2.5 |
| png 1.5.1 | CDAT 1.3.1 | jpeg v8c | tiff 3.9.4 | pbmplus |
| Tlc/Tk 8.5.9 | LAPACK/CLAPACK | freetype 2.4.10 | Pixman 0.30.0 | fontconfig 2.10.1 |
| Cairo 1.10.2 | uuid 1.6.2 | udunits2 2.1.24 | shapely 1.2.14 | HDF5 1.8.8 |
| NetCDF 4.3.0 | Qt 4.7.2 | jasper 1.900.1 | g2clib 1.2.5 | Python 2.7.4 |
| VisIt 2.6.0 | SIP 4.14.6 | PyQt 4.10.1 | PyOpenGL | Numpy 1.7.1 |
| Pmw 1.3.2 | CMOR 2.8.3 | Matplotlib 1.2.0 | GEOS 3.3.5 | Libcf 1.0-beta11 |
| Cython 0.16 | MPI 1.6.4 | ESMF 6.1.0 | setuptools 0.6c11 | gui_support |
| distribute 0.6.45 | Pip 1.3.1 | MyProxyClient 1.3.0 | Numexpr 2.1 | Lepl 5.1.3 |
| Sphinx 1.2.b1 | pyzmq 13.1.0 | spyder 2.2.0 | tornado 3.1 | IPYTHON 0.13 |
| Mpi4py 1.3 | NetCDFPLUS 4.2.1.1 | R 2.15.1 | ParaView 3.11.1 | Pyspharm 1.0.7 |
| PyTables 2.4.0 | SCIPY 0.12.0 | scikits 0.12 | VTK 5.9.0 | VisTrails |
| DV3D | CDATLogger | WGET 1.12 | gdal 1.9.1 | docutiles 0.10 |
| jinja2 2.7 | pyopenssl 0.13 | pygments 1.6 | blaze | llvm |
| llvmpy | Django 1.5.1 | Numba 0.9 | | |

The UV-CDAT team used Git, an open-source distributed version control system, to manage the UV-CDAT software repository. Git encourages branch workflows where each feature can be a separate branch. If there are multiple developers working on a single project, which is typically the case for UV-CDAT, it is essential to have a workflow that facilitates collaborative development and reduce software breakages. The UV-CDAT software development team set up three maintenance branches: "master," "next," and "release." "Master" is the most reliable integration branch, "next" is the cutting-edge integration branch, and "release" is the latest release branch. This separation allowed the team to build and test each of these branches on dashboards to ensure software quality and reliability.

Kitware CMake was used for building software. Kitware CTest is a test-driver tool, used to run regression tests, and Kitware CDash is a web application for displaying testing results on dashboards and performing continuous integration testing. UV-CDAT primarily uses CMake and other Python scripts to build libraries and their dependencies on any given system. To ensure that UV-CDAT tools and applications build and perform as expected, it was essential that the tools be built and that regression tests were run when developers pushed new changes to the repository. The development team worked towards setting continuous (i.e., run when the repository server receives new changes) and nightly (i.e., runs every night at a predefined time) dashboards, which built and tested UV-CDAT against the latest changes.

### *3.3.3 Simple GUI*

The UV-CDAT GUI, the main window for UV-CDAT, is shown in Figure 4. It is based on the notion of a visualization spreadsheet (middle) or a resizable grid in which each cell contains one or more visual images. By using intuitive drag-and-drop operations, visualizations can be created, modified, copied, rearranged, and compared. Spreadsheets maintain their provenance and can be saved and reloaded. These visualizations can be used for data exploration and decision-making, while at the same time are completely customizable and reproducible.

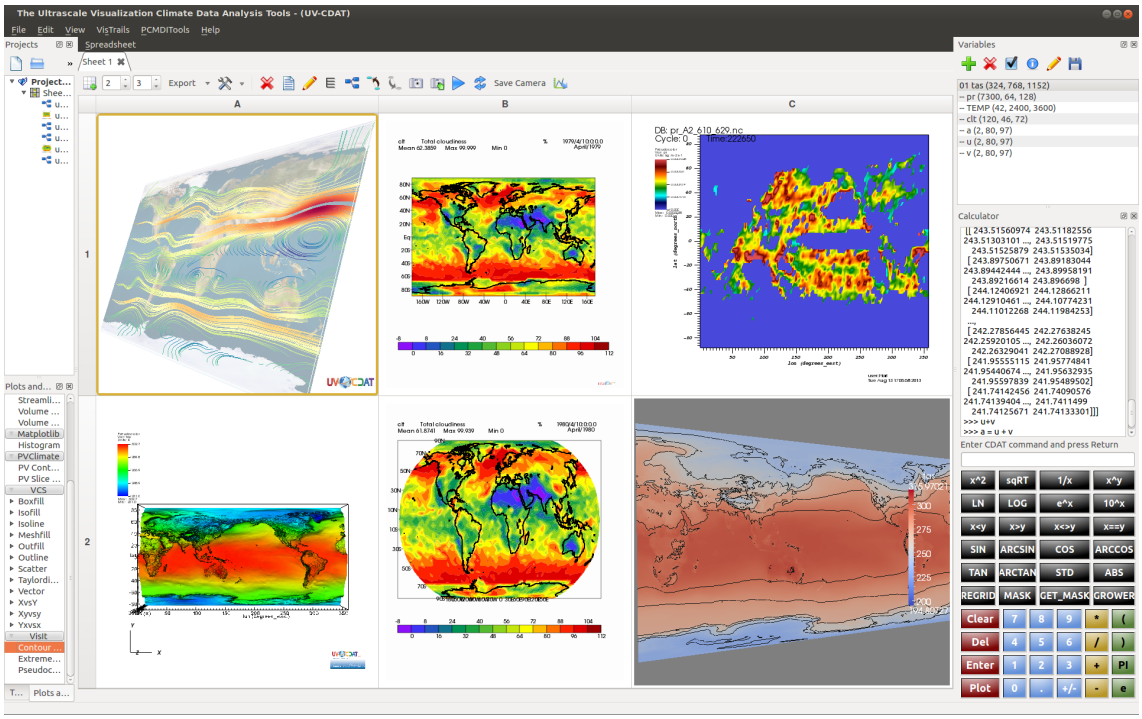



***Figure 4.*** *In the UV-CDAT GUI, extreme value analysis plotting can be accomplished using an array of visualization tools, such as VisIt-R (top left), ParaView plot (bottom left), CDAT plot (top and bottom center), DV3D (top right), and VisIt (bottom right).*

Around the spreadsheet are the tools for building visualizations, as shown in Figure 4. The project view (top left) allows a user to group spreadsheets into projects and to name visualizations and spreadsheets. The plot view (bottom left) allows for the use and customization of available plot types. The variable view (top right) allows data variables to be edited. The bottom right contains a variable editor widget that makes editing a variable much like using a pocket calculator. With the UV-CDAT GUI, creating a visualization can be as simple as dragging a variable to a spreadsheet cell and then dragging a plot type to the same cell.

### 3.3.4 Successful Use of DV3D

Using the three-dimensional DV3D tools in UV-CDAT, users are able to model and explore observational data in ways that previously would have been difficult if not impossible. The use cases described below—two of many important use cases—demonstrate the success of DV3D:

- ***Meridional (V) wind anomalies at 200hPa.*** Using the Hovmöller slicer on the MERRA reanalysis, an interesting pattern emerged in the mid-latitude 200hPa V wind anomaly. The UV-CDAT software climate scientist team have not yet identified the source of this apparent wave that appears to have a period of about five years, and it may be a feature of the input data to reanalysis. The pattern seems to appear after 1995 and may be associated with stationary Rossby waves. We are still investigating this phenomenon, and while it may be an artifact of data assimilation, it may prove to be important.
- ***Details of jet stream location from AMWG diagnostics.*** The Atmospheric Modeling Working Group (AMWG) diagnostic package generates nearly 600 plots for one experiment. The plots are displayed in two dimensions (e.g., latitude–height), but it is impossible to see any zonal regional differences. By invoking UV-CDAT/DV3D, it is possible to examine in detail the regional differences that occur in the zonal (U) wind component revealing subtle difference in the zonal winds and shifts in the pole-ward movement of the mid-latitude jet stream.

In order to facilitate use case exploratory processes, the NCSS has deployed UV-CDAT within a distributed parallel visualization framework to enable simultaneous interactive visualization of the scores of variables involved in many model diagnostic and intercomparison tasks. This framework employs a hyperwall cluster that currently consists of a 5x3 array of 46-inch touchscreen displays, each with a dedicated compute (client) node, plus a single control (server) node, as shown in Figure 5.

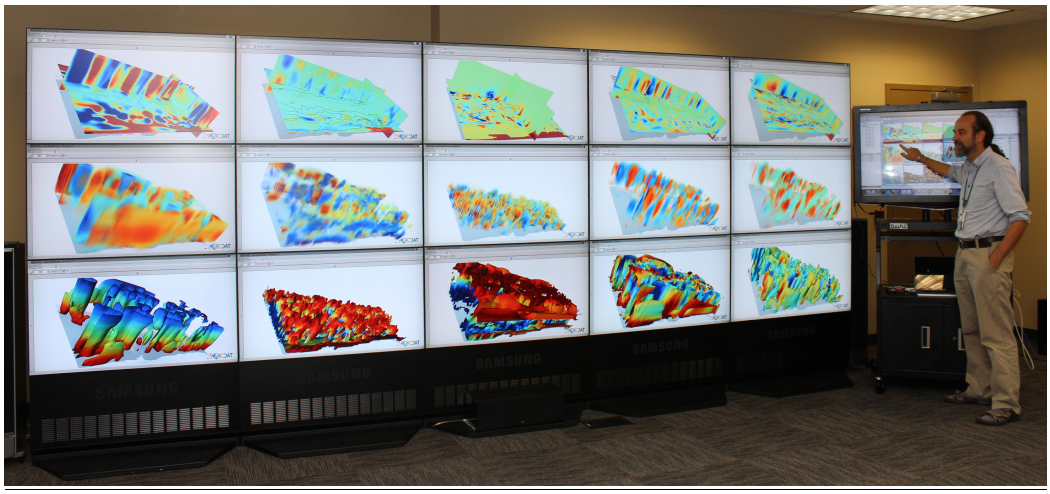


***Figure 5.*** *The UV-CDAT distributed visualization framework deployed on the NASA NCCS hyperwall. UV-CDAT team member Tom Maxwell uses the touch screen to demonstrate the interactive procedure. This configuration enables scientists to exploit the full capacity of the hyperwall cluster (32 dual-core Xeon Harpertown processors, 16 Quadro FX 1700 GPUs, and a 17- by 6-foot, 15.7-million pixel display) in support of interactive visual data analysis and understanding of very complex climate data sets.*

In this framework, an instance of UV-CDAT runs on each node, coordinated using socket connections between the client nodes and the server node. Each client instance opens a single cell visualization spreadsheet window, covering its hyperwall display. The user interacts with the GUI of the server instance using one of the touchscreen displays. In a typical use case scenario, the user opens (or constructs) a workflow with 15 cell modules on the server node. At execution time, the server instance sends edited versions of the workflow to each client node for local execution. Each client workflow consists of one of the cell modules (and all its upstream modules) from the server workflow.

### 3.3.5 UV-CDAT Spatio-Temporal Parallel Pipeline using ParaView

Many climate data sets have not only a high spatial resolution but also a high temporal resolution. In many instances, simulations output daily or monthly averages, with data sets possibly covering a total timespan of decades. Due to this high temporal resolution, there is a critical need to have a fast, scalable method for processing a large number of timesteps. In response, the UV-CDAT spatio-temporal pipeline was devised to solve UV-CDAT Use Cases 1 and 2.

***Use Case 1:*** Use Case 1 involves a data set with high spatial and temporal resolution. The task is to produce an image sequence of the data set by producing one image per timestep. A key aspect is that there are no dependencies between timesteps, and each timestep can be processed independently.

In a fully data-parallel model, all available processors take part in reading the first timestep, perform any required analysis, and then take part in producing the final image output. Then all processors load the second timestep, and the cycle repeats until all timesteps have been processed. In this case, timesteps are accessed sequentially, and only spatial parallelism is utilized.

In the UV-CDAT spatio-temporal pipeline, parallelism is extended both spatially and temporally, allowing multiple timesteps to be processed simultaneously. This is permissible since each timestep can be processed independently of all others. In order to achieve temporal parallelism, the available processors are divided into groups called time compartments. The number of total time compartments is dependent on the number of processes and the size of a time compartment, which is constant over all time compartments. Each time compartment can now load and process a timestep independently of all others, resulting in multiple timesteps being worked on simultaneously.

Figure 6 illustrates how processors are broken up into time compartments and how work is distributed among them. When there are more timesteps than time compartments, the files are assigned in round-robin fashion to time compartments, so it is possible for a time compartment to process multiple timesteps. Within a time compartment, its assigned timesteps are processed one at a time.

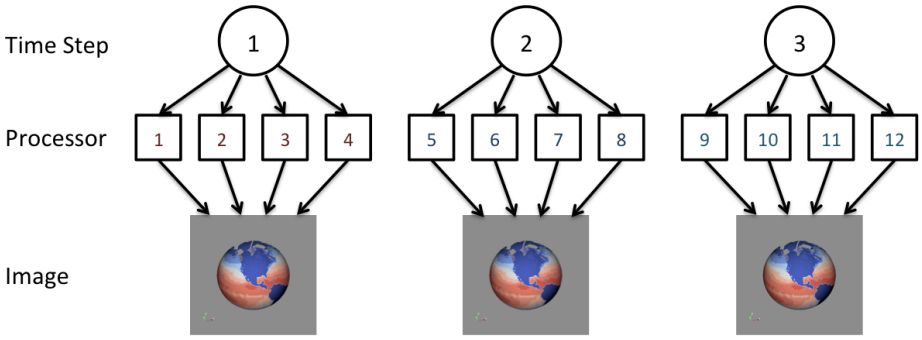



***Figure 6.*** *A high-spatial resolution and high-temporal resolution and image sequence production use case is handled by multiple time compartments, each processing many time steps simultaneously.*

The amount of temporal parallelism versus spatial parallelism is controlled by the size of one time compartment, which is a user configurable parameter. When the time compartment size is large, there are fewer total time compartments, which results in less temporal parallelism and more spatial parallelism. On the other hand, if the

time compartment size is small, there are then more time compartments, and the amount of temporal parallelism is higher, the amount of spatial parallelism decreases.

***Use Case 2:*** Use Case 2 also involves analyzing climate data that has both high spatial and temporal resolution. The difference is that instead of having each timestep processed independently, some type of reduction operation among several timesteps must be performed, with the output being various statistics. One example would be calculating yearly averages from monthly data. In this case, the 12 timesteps comprising one year must be averaged together. Supported reduction operations include average, minimum, maximum, and standard deviation.

The UV-CDAT spatio-temporal pipeline can also be used to support Use Case 2, as illustrated in Figure 7. Each time compartment loads one or more timesteps as in Use Case 1, but with an additional reduction step, which combines the results of several time compartments into a single data product.

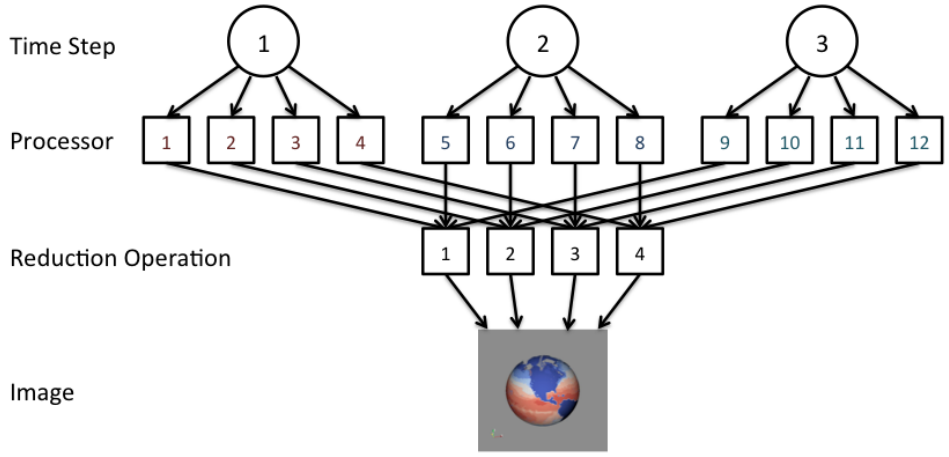



***Figure 7.*** *For a high-spatial resolution, high-temporal resolution, and time-average use case, an additional reduction step is necessary to achieve a single data product.*

Initial results from the UV-CDAT spatio-temporal pipeline's performance show vast performance gains. In Use Case 1, there was an improvement from hours (using one processor doing sequential steps) to minutes when switching to the spatio-temporal pipeline. Use Case 2 also showed an order-of-magnitude improvement when using the spatio-temporal pipeline. Similar results have been shown over multiple supercomputers and multiple file systems.

### *3.3.6 Exploratory Data analysis ENvironment (EDEN)*

The Exploratory Data analysis ENvironment (EDEN), shown in Figure 8, offers a unique approach to exploratory data analysis of multivariate data. EDEN adopts a visual analytics approach to climate analysis with a parallel, coordinates-based canvas augmented with information from automated statistical analytics. EDEN supports dynamic visual queries with multiple coordinated views that effectively guide the scientist to the most significant associations in the data. EDEN provides an information visualization-centric capability within the context of the broader suite of scientific visualization and analysis tools provided by UV-CDAT.

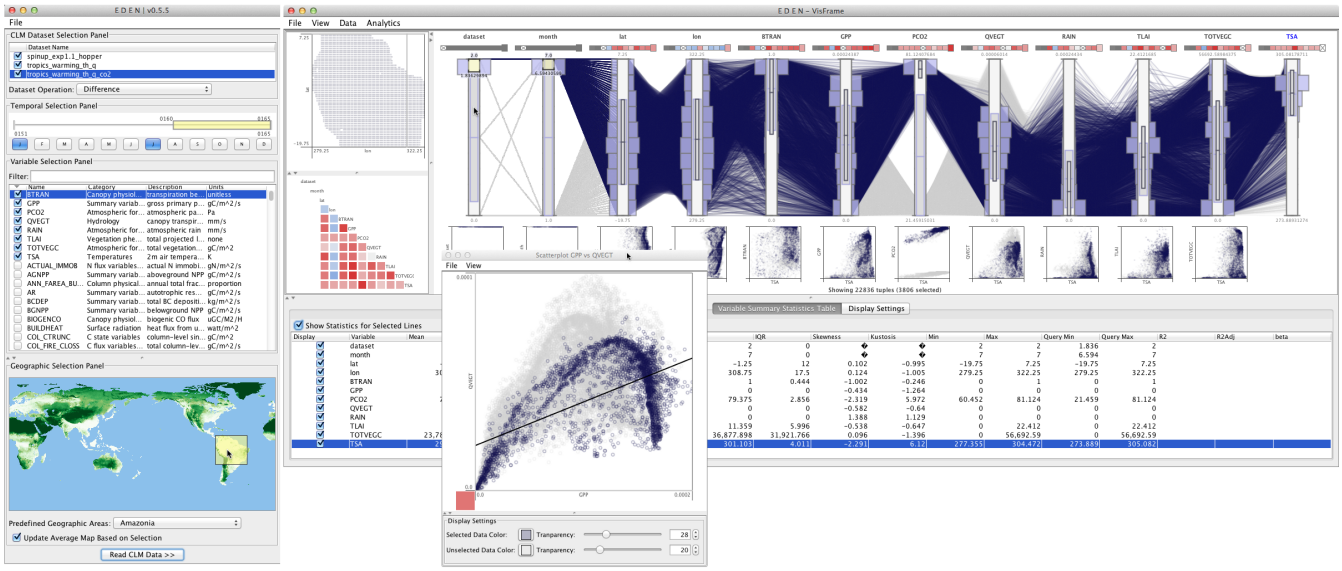


***Figure 8.*** *EDEN provides an overview during analysis of a global Community Land Model (CLM) data set. The CLM filter panel (left) facilitates interactive queries into large CLM data sets. The VisFrame (right) offers a highly interactive, visual interface to explore multivariate relationships via linked parallel coordinates, scatterplots, a correlation matrix, and geographic scatterplot visualizations.*

EDEN fosters interactive visual queries via several graphical techniques. The Community Land Model (CLM) filter panel offers user interface components for visually forming multifaceted selections. The process uses information sent to assist the scientist in finding the most promising relationships for detailed investigation. The panel provides widgets for dynamic queries in terms of simulation, temporal ranges, variables, and geographic region of interest. When the filter criteria are selected, the scientist can read in the data from the simulation(s) into EDEN's VisFrame for detailed investigation.

The VisFrame interface is an interactive multivariate visual analysis canvas that is composed of a number of inferential information visualization techniques. The views are connected in a coordinated model so that changes and filter range selections are propagated to the various views appropriately. The VisFrame is built around a highly interactive variant of the parallel coordinates visualization technique. Common sets of parallel coordinate features are available in EDEN, such as movable axes, details-on-demand, polyline brushings, and automated axis arrangements. Furthermore, the parallel coordinate plot is extended with statistical analytics that automatically augment the display for guided analysis.

For example, each vertical axis in the parallel coordinates canvas represents a variable selected in the CLM filter panel. The axes are augmented with visual cues that inform the scientist's exploration of the information space. A scientist can rapidly build visual queries by brushing regions of the axes to select lines of interest, which represent multivariate tuples of the selected data. In this way, multiple queries on separate axes are used to construct conjunctive selections. Selected polylines are shaded more prominently to boost their visual salience, and certain key descriptive statistics are represented graphically in the interior of each axis. For example, embedded box plots communicate the mean and standard deviation range for the overall distribution.

For detailed analysis, a user can access a panel of scatterplots to explore additional details such as nonlinear trends, thresholds, and clusters. EDEN facilitates visual correlation mining to judge the strength of interrelated variables and visually highlight significant associations. The correlation statistics are updated, based on user interactions, and used to augment the displays. A correlation matrix is formed in memory and graphically represented in a display of interactive color-coded blocks. The blocks are encoded with color to indicate the type (blue for negative and red for positive) and strength of the correlation (stronger correlations receive more saturated colors).

We have evaluated EDEN in close collaboration with CLM researchers. EDEN improves on the prior CLM diagnostics package by providing not only new multivariate views of the data but also interactive exploration of the parameter space with dynamic visual queries. We executed two case studies with climate scientists to demonstrate the advantage of a visual analytics approach in a global simulation case study and a smaller point-based ensemble analysis. These case studies involved real-world simulations, and the climate scientists who are also co-authors of the subsequent publications drove the analysis.

### *3.3.7 ParCat*

Much of the post-processing and analysis work involved in studying climate data sets requires fast statistical methods—for calculating averages and histograms, and finding minimums/maximums over a data set. Additionally, multi-model comparison runs require the difference of two data sets or finding an average-of-differences. These functions accelerate use cases 3, 4, and 8 identified in section 3.2.

ParCAT, which was developed specifically for UV-CDAT, can improve time-to-solution for analysis of data sets with high temporal resolution or number of timesteps but with lower spatial resolution. ParCAT can provide averages, differences, or sums of two data sets, histograms, and the average-of-differences of two data sets, and can find the minimum and maximum values for a data set. Many of these statistical operations are embarrassingly parallel in multiple dimensions. For example, calculating a spatio-temporal average map is parallelizable in time, space, and per variable. Some variables even allow for a fourth parallelizable dimension—height or depth. However, because these data sets comprise hundreds of files, consuming hundreds of gigabytes of storage, the amount of I/O (relative to compute time) is a critical factor for achieving good parallel speedups.

ParCAT reads standard network Common Data Format (netCDF) files, produces netCDF output, and uses parallel netCDF libraries to read and write the files in parallel. ParCAT is also a command-line utility that can be built on a wide variety of platforms. This makes it easy to embed ParCAT in other analysis tools such as UV-CDAT or in scripts.

A front-end GUI was developed for ParCAT so parallel (pre-) processing of large data sets can be accomplished directly from within UV-CDAT. The user selects which data sets are of interest and what processing needs doing, then the job is submitted and the results are returned to UV-CDAT.

Because most processors or nodes utilize multiple compute cores for processing, ParCAT was designed to use both multiple cores on a node and multiple nodes to maximize I/O efficiency. Each node opens first one or more files and then each core on the node processes data. In this way, there is less I/O contention, and ParCAT will scale to at least one node per file in the data set.

Figure 9 shows results of processing a data set using a small partition on the Titan supercomputer. There are 349-variables with ½-degree spatial resolution and 20 years of monthly timesteps (240 files total). ParCAT processes multiple variables in parallel across cores and reads multiple files per node. With a larger data set and more nodes, the speedup from one process should be even better.

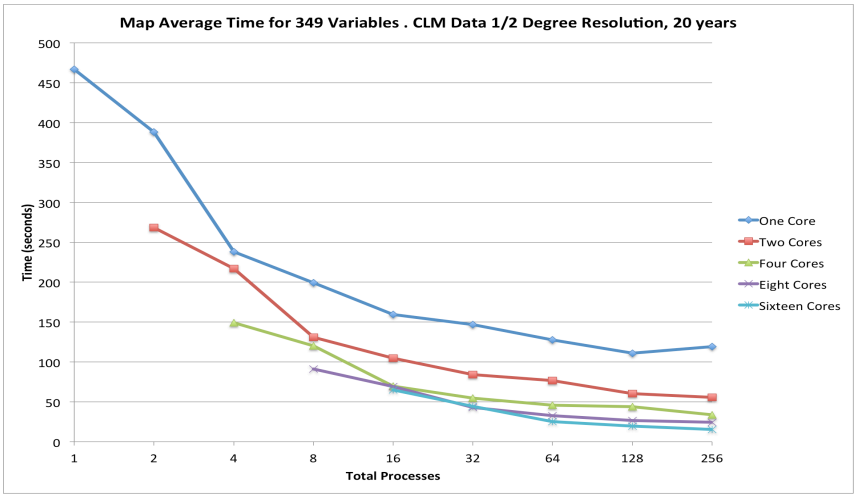



***Figure 9.*** *Using ParCAT on Titan, with from 1 to 16 processing cores calculating the temporal average of 20 years of monthly data with 349 variables.*

### *3.3.8 Distributed Arrays*

Use cases 2–8 can benefit from using distributed arrays as ParCAT does, by leveraging multiple processing units to perform similar post-processing tasks on different sets of data, faster, and with a smaller memory footprint per processor. However, distributed arrays can also be applied to cases where each processor (or core) must communicate with other processors, not only with a master processor. Data filtering operations (use case 13), including smoothing and feature extraction, fall in this category.

ParCAT's limits to parallel scalability of distributed arrays include the ratio of serial to parallel section (not all operations can be parallelized) and load balancing (which can cause some processors to wait for other processors). Both of these effects cause the parallel speedup—the ratio single processor wall clock time to parallel processor execution time—to plateau beyond a certain number of processors. For distributed arrays, there is also the communication cost, which itself involves latency and throughput, and which increases with higher numbers of processors. As a result, the execution speed always decreases beyond a certain number of processors when there is intra-processor communication. To reduce the communication cost, the partitioning of the data should always attempt to minimize the ratio of surface (proportional to communication) over volume (proportional to the work load).

UV-CDAT implements a very flexible distributed array representation based on Message Passing Interface 2 (MPI-2) one-sided communication. As with other distributed array implementations, each processor holds a subset of the data. However, in contrast to other implementations, subsets of the local data can be exported to any other processor, not only to a neighboring processor. Moreover, UV-CDAT supports distributed arrays in any number of dimensions for all the common data types (float, double, and integer). We achieved this by breaking the local data array into sub-boxes, with each sub-box uniquely defined by a set of integer indices, regardless of the dimensionality of the array.

A particular challenge was to integrate the distributed array capability into software that was originally designed to operate serially. Because there is a large body of legacy CDAT scripts, the new distributed array extensions were required to operate seamlessly with older scripts. We opted for a minimalist approach, adding a very small set of methods to the Climate Data Management System (CDMS) array object, which is essentially a standard Numpy array object with additional metadata and methods attached. The new distributed arrays methods in CDMS are:

- exposeHalo(ghostWidth)
- fetchHaloData(procNumber, windowId)

Here, ghostWidth refers to the number of ghost values (the same in all directions), procNumber the processor rank, and windowId the unique sub-box identifier for each data window (e.g., the tuple [-1,0] would denote the west side of the array). The Python implementation uses the Mpi4Py module to exchange data among processors. Despite its simplicity, this application–programming interface supports discretization filters of any order. Figure 10 shows the scaling obtained by applying a two-dimensional finite difference Laplace operator discretization on a 128-core cluster to extract features (the Laplacian is highest where the data have sharp gradients). The size of the local data (2000x2000) was kept constant as the number of processors increased (weak scaling). The communication cost increases and amounts to 15% for 128 processors. Therefore, a 128-times-larger problem can be solved using only 15% more wall clock time.

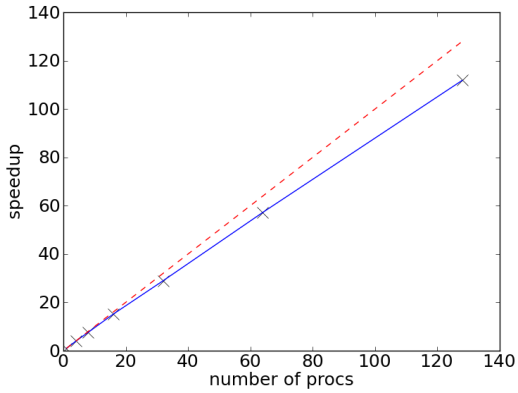



***Figure 10.*** *Good scaling of UV-CDAT's distributed arrays is applied to a two-dimensional Laplace filter post-processing test case. The red dashed line represents the ideal scaling. The blue solid line represents the actual speedup obtained on a 128-core cluster with Infiniband interconnect.*

### *3.3.9 VisIt*

VisIt's server infrastructure provides parallelism in three significant categories: databases, plots, and operations. These three components are individually optimized to take advantage of scaling and communication requirements with respect to underlying algorithms. Databases provide the ability to read massive data sets in parallel, where operations can manipulate the data on a per-process basis and finally visualize the plot using a scalable rendering functionality. VisIt's integration into UV-CDAT enables climate science users to exercise several spatio-temporal parallel processing algorithms described below.

Parallel databases are aware of the underlying representation on the disk and can take advantage of domain decomposition and parallel I/O operations when needed for analysis and visualization plots. In addition, VisIt's operations are aware of the decomposition and are able to operate over the data utilizing communication to solve the task at hand. In addition, all plots within VisIt support parallel rendering in scalable rendering mode to handle visualization needs even for the largest data sets.

New parallel analysis frameworks were added to VisIt, including extensions to the current set of operators. Generalized Extreme Value Analysis and Peaks Over-Threshold are two common operators exercised frequently by the climate science community. Figure 11 portrays both operators processing more than 20 years of information and correlating two separate ensemble runs. We exercised several custom R analysis routines within VisIt to provide a robust solution to the climate community. Both operators exercise spatio-temporal parallelism.

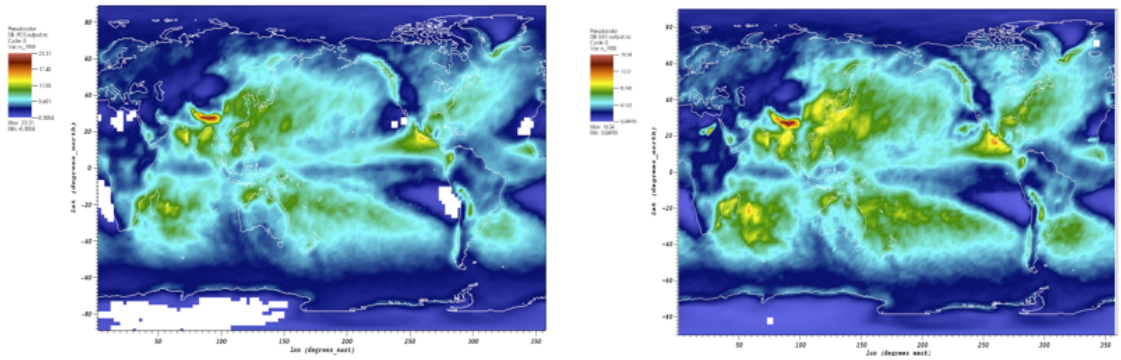


***Figure 11.*** *20-year annual return values for daily precipitation CAM5.1, 1979-2005, ensemble of 2 runs are shown in Peaks-Over-Threshold (left) and Generalized Extreme Value Analysis (right).*

Although the addition of optimized VisIt operations provides a new range of capabilities to our users, scientists still requested modifications greater than the parameter space allowed by the algorithms. Users also wanted to exercise familiar functionality such as from the CDAT toolset. Our solution, and the second parallel analysis extension, was to add a general scripting operator within VisIt to facilitate creating custom analysis routines in addition to pre-existing operations. The new scripting framework in VisIt supports Python & R. We also provided helper functions to hide complex requirements for communicating in a distributed environment, as well as a mechanism to import the core CDAT application programming interface (API) into VisIt, thereby allowing users to exercise climate-centric functionality in an embarrassingly parallel way. Furthermore, we included MPI support in Python & R to create communication among custom scripts written by the user community.

Figure 12 shows VisIt's ability to easily create variants of analysis operations with very simple changes. Several core features of the parallel analysis framework include template function support, inclusion of familiar climate tool-chains such as CDAT, and access to parallel libraries such as Mpi4Py in Python & pbdMPI in R.

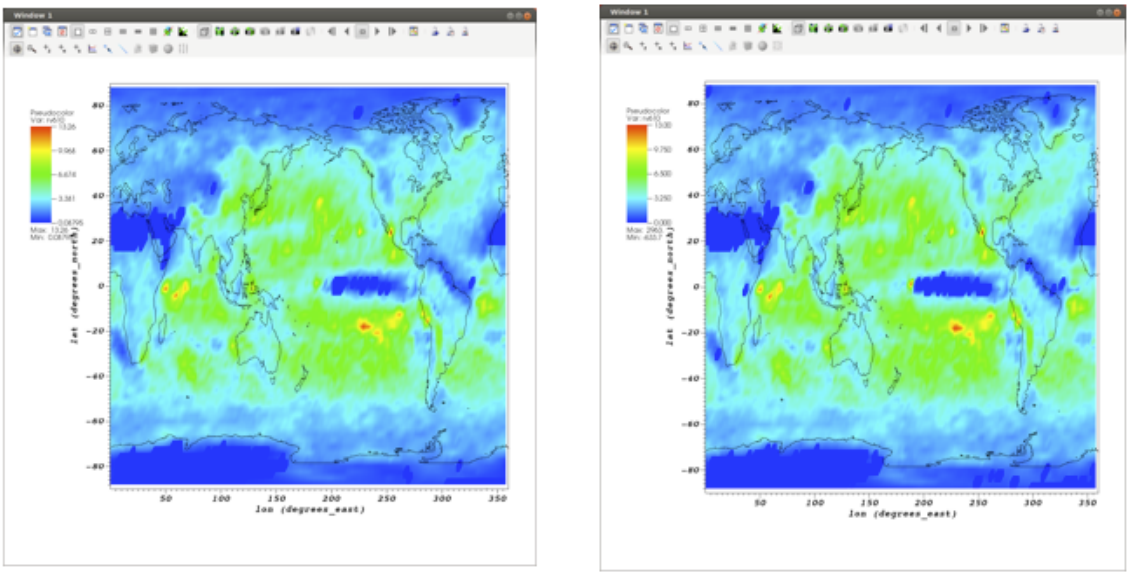


***Figure 12.*** *VisIt can be extended to perform dynamic and complex operations in VisIt using Python & R. Two variants of analysis operations are shown.*

The third analysis addition to VisIt includes a new set of tools to enable parallel climate science analysis. The stand-alone deliverable provides analysis routines that characterize tropical cyclones across models (CAM5), resolutions (2,1,1/4,1/8 degrees), and forcings (climo, SST+2, 2xCO2). The code-base also evaluates precipitation and cyclogenesis. In addition, our team provided algorithms for detecting extra-tropical cyclones, atmospheric

rivers, and extreme precipitation structure. These algorithms are wrapped within a common infrastructure called TECA (Parallel Toolkit for Extreme Climate Analysis) that was written with MPI support.

### 3.3.10 ViSUS

The ViSUS—Visualization Streams for Ultimate Scalability—framework developed at the University of Utah Center for Extreme Data Management, Analysis and Visualization and LLNL has been designed to enable interactive visualization and analysis of massive data. Using a novel data layout, ViSUS enables users to quickly and effortlessly browse some of the largest scientific data sets ever created. Furthermore, the system scales well across machines as well as across resources, allowing data access from simple mobile devices such as tablet computers all the way to high-end workstations and from low-end WiFi connections to high-end dedicated networks.

In the context of climate analysis and UV-CDAT specifically, ViSUS addresses a significant need to quickly and efficiently explore large amounts of remote data. Unlike even a decade ago, climate data can no longer be collected and stored in a central location but instead must be kept in large federated, distributed archives accessible, for example, through the ESGF. Nevertheless, remote analysis capabilities are still in their infancy. Thus, a typical workflow consists of finding data sets of interest, identifying the subsets relevant to the analysis of interest, downloading the data, and finally analyzing it locally to gain new insights. In the past, scientists typically downloaded entire simulation ensembles or databases to avoid having to process remotely stored data. However, given the explosion in data size, this scenario is no longer feasible. Instead new tools are needed to enable scientists to decide which data is most relevant and necessary for any particular analysis.

The ViSUS plugin for UV-CDAT provides these capabilities for the large ensembles of climate simulations that are part of, for example, the next IPCC report. As shown in the Figure 13, the ViSUS plugin enables users to select a remote server that hosts the data and interactively explore the various, simulation runs, attributes, and time steps using a simple interface. The system is based on a simple re-ordering of the data, which we currently perform *a priori,* but which we envision will be handled on demand as data is accessed in the future. Furthermore, as discussed in more detail below, we, together with the ViSUS team, have developed a new lightweight remote analysis toolkit that enables users to easily integrate a number of common operators, e.g. means, standard deviations, etc., into the remote exploration. These operators will be executed on the server-side, allowing users to remotely visualize and analyze derived products such as ensemble means.

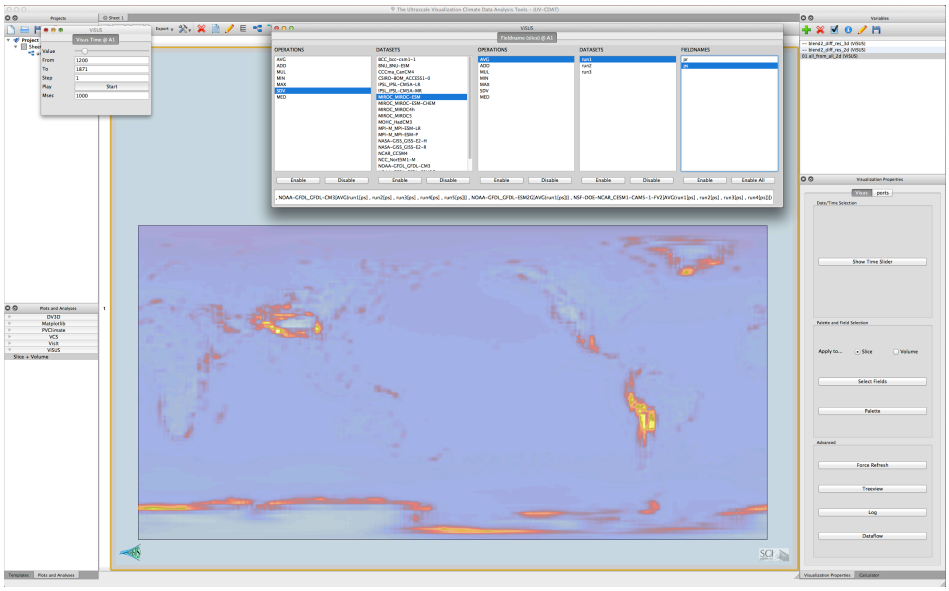


***Figure 13****.ViSUS plugins enable users to select remote servers that host data.*

One of the most common operations in climate science is to aggregate simulation ensembles by creating, for example, the ensemble mean. However, the process requires not only significant time and compute resources but also assumes that the source data is available at a central location. Given the current and expected future data sizes, downloading and processing simulation ensembles is a daunting task that can consume large numbers of man-hours. The problem is compounded by the fact that an ensemble may contain artifacts or outliers, which can easily skew the results. However, removing, for example, an outlier calculation currently requires the user to re-compute all aggregated data, which leads to a tedious and resource intensive workflow. Instead, ViSUS, and by extension UV-CDAT, now provide the ability to create such ensemble aggregations on-the-fly in a server-side manner that is completely transparent to the users.

As shown in the Figure 14, the user is presented with all simulations available at a remote site and can use any subset of simulations and or variables in a dynamically created expression, for example, to compute the mean or standard deviation of an ensemble. The system will evaluate the query server-side and deliver the resulting data to the local client for visualization and analysis. At any point, the user can change the expression as well as the source data to quickly explore different ensembles, identify outliers, and determine the most useful (sub)set of data for the analysis of interest. In this manner, ViSUS allows users to preview the results of any large-scale aggregation in an interactive and highly flexible manner. In the future, we envision that these capabilities will significantly reduce the amount of data required locally for analysis by providing many of the standard capabilities through remote, streaming processing and visualization.

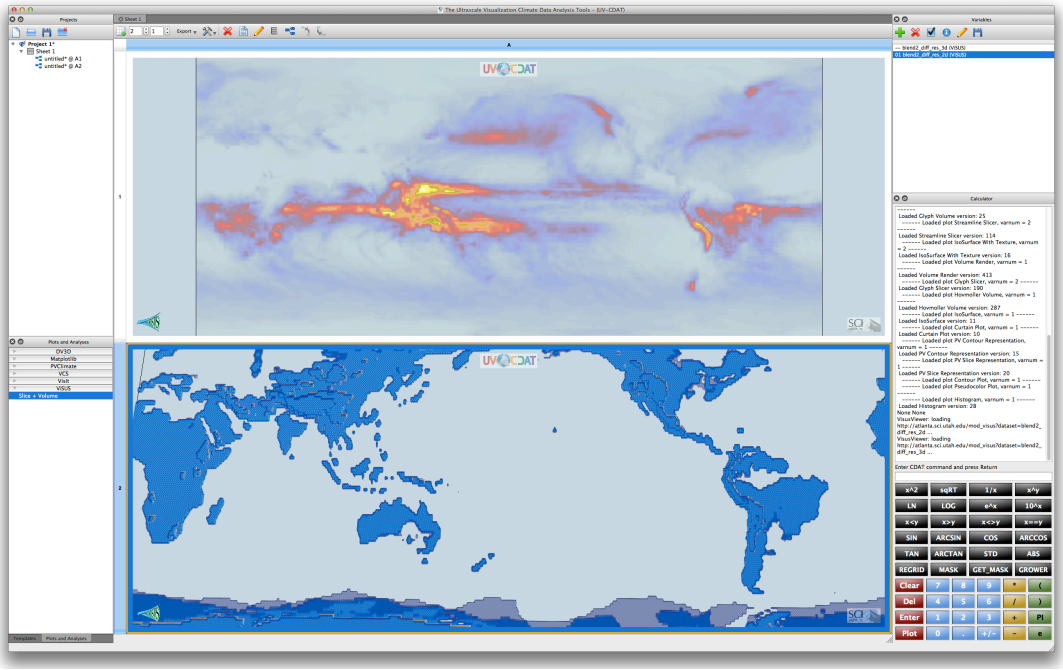


***Figure 14**. ViSUS interactive visualizations shown through the UV-CDAT interface.*

## 3.4 Visualizing Weather and Forecasting (WRF) Model Data with UV-CDAT

UV-CDAT has been extended to visualize data from the Weather Research and Forecasting (WRF) model, either in model coordinates or regridded to latitude-longitude coordinates, as shown in Figure 15. Using the built-in conservative regridding capabilities, WRF data can be subselected and downloaded from an OPeNDAP server. A parallelized CDAT routine then regrids the data for use in UV-CDAT. The data can be visualized (as shown in Figure 15 ) in two or three dimensions and includes 3D animation.

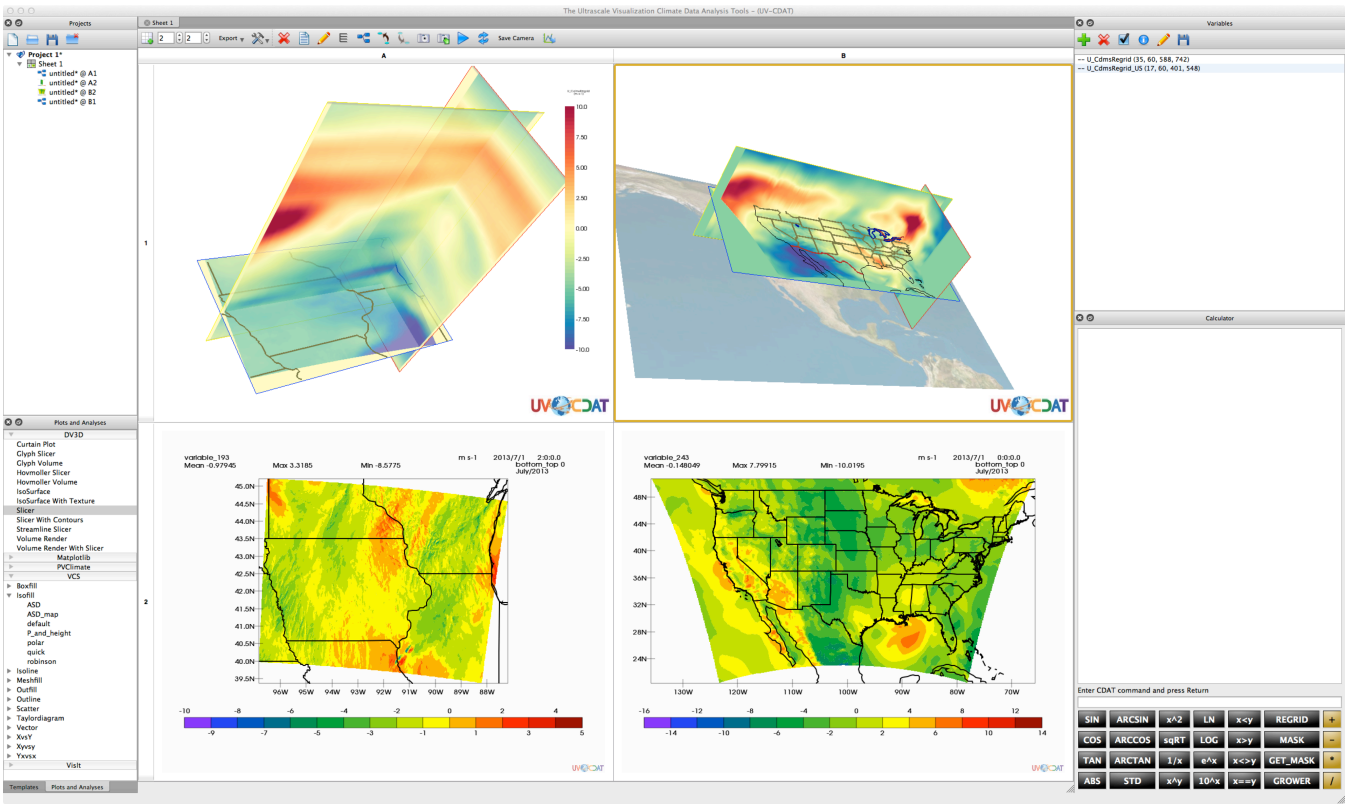


***Figure 15.*** *WRF model output displayed in 2D using CDAT's VCS and 3D using DV3D.*

## 3.5 Point Cloud Viewer

The purpose of the new DV3D point cloud plotter (under development) is to allow scientists to interactively construct 3D views of climate data in its raw (unprocessed) geometry. The current DV3D plotters require the data to be laid out on a rectangular grid. If the input data is on an irregular or curvilinear grid, then it must be resampled and interpolated to a regular array. However, this regridding process is undesirable in many cases because it can distort the data. For example, a scientist may be looking for computational artifacts at the tile boundaries of a cubed-sphere grid, as shown in Figure 16. The process of regridding the data can warp and mask the features of interest.

The new DV3D point cloud plotter allows this data to be viewed directly without requiring any preprocessing. It makes no assumptions regarding the geometrical layout of the points, visualizes the points directly, and colors each point by the value of the variable at that location. Using this display method it is very easy to shift projections (e.g., toggle between a latitude-longitude level rectangular projection and a spherical projection). By selectively filtering and varying the opacity of the points, the user can generate a wide range of slices, isosurfaces, and volume renderings. The same filtering process is applicable to data that are gridded (i.e., regular, curvilinear, or unstructured) or ungridded. These disparate data layouts are all easily overlaid in a single view. The current version of the viewer has been tested with data from the following models: Community Atmosphere Model (CAM), Goddard Earth Observatory System Model, Version 5 (GEOS5), European Center for Medium-Range Weather Forecasts (ECMWF), WRF, and Multi-scale Modeling Framework (MMF).

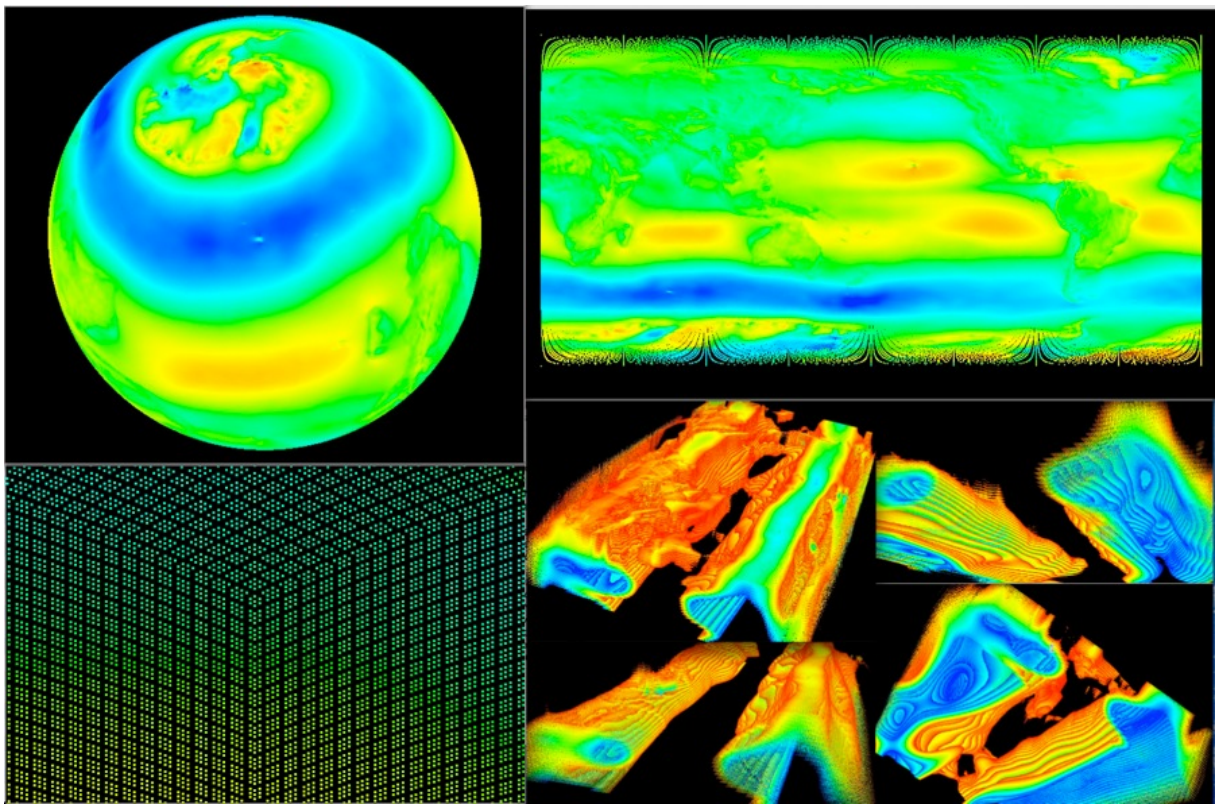


*Figure 16. (Top-left) Cubed-sphere CAM data, sliced along constant z-coordinate, in spherical projection; (top-right) cubed-sphere CAM data, sliced along constant z-coordinate, in latitude-longitude projection;( bottom-left) cubed-sphere CAM data zoomed-in to reveal points; and (bottom-right) cubed-sphere CAM data volumes generating by thresh holding on a range of variable values.*

## 3.6 Administration, Operations, and Plans for Maintenance

Management of UV-CDAT is critically important for the future service of high-profile (and other equally important if lower profile) analysis and visualization of large-scale data sets. From our experience with disparate development and as a collective, we know that future UV-CDAT management must address five distinct functions that are the common building blocks for any large software effort:

- **Operations** to ensure effective implementation and control of such activities as the status and monitoring of hardware, network, and software systems to ensure secure and reliable processing operations and data analysis, visualization, and diagnostics.
- **Maintenance** to ensure effective implementation and control of software, hardware, and network activities to provide optimum performance levels needed by the community.
- **Engineering** to ensure effective implementation and control of technical support. This work entails proper design review, required development to conform to new standards and technologies, and documentation of changes for others to follow.
- **Training** to ensure effective implementation and control of activities and user support, including the training necessary for new projects, data providers, and the diverse user community. It also includes the UV-CDAT help desk, Frequently Asked Questions (FAQs), software-use tutorials (e.g., the UV-CDAT Tutorial: http://uv-cdat.org/tutorials/), support mailing lists (such as the uvcdat-support mailing list), and websites and wiki sites (such as the uv-cdat.org site).

- **Administration** to ensure the establishment of policies and the planning and control of activities. This work is especially necessary for the overall operations, monitoring, and direction of the UV-CDAT performance and interfacing with stakeholders, program sponsors, team members, data centers, and programs.

For all these distinct functions, user knowledge and performance will help to support a reliable UV-CDAT framework.

UV-CDAT will never be a static system. As the platforms on which it operates—server hardware, networks, operating systems, and browsers—evolve, UV-CDAT software must be adapted. UV-CDAT is also a collaborative development with components from several quasi-independent projects. As one component advances, adaptations in others will be required to support the updated functionality.

## 3.7 Help Desk

Our partners and the UV-CDAT team established a community help desk to assist users with information and resources and to help troubleshoot problems with analyzing and visualizing data, the UV-CDAT ecosystem, and products. This help desk service is provided in the form of software, website, and e-mail. Traffic on the help desk is increasing significantly with increases in recognition and use. Figure 17 shows the new Askbot website where users ask questions and create help tickets. UV-CDAT team members are essential to keeping this operation going. We anticipate considerable traffic of user discussions with help desk providers and follow-on questions by other users.

Scientists at DOE and NASA are charged with addressing science questions, while technical staffs at these institutions are charged with addressing UV-CDAT framework questions. These personnel spend a portion of their day scanning the list of new questions, divvying up the workload, and responding to users. Questions that are resolved are placed on the UV-CDAT FAQ list to circumvent answering repeat questions. We are moving away from the uvcdat-support mailing list to the Askbot help desk for greater control and better tracking of questions. To do this, we are redirecting users from uvcdat-support mailing list to the help desk.

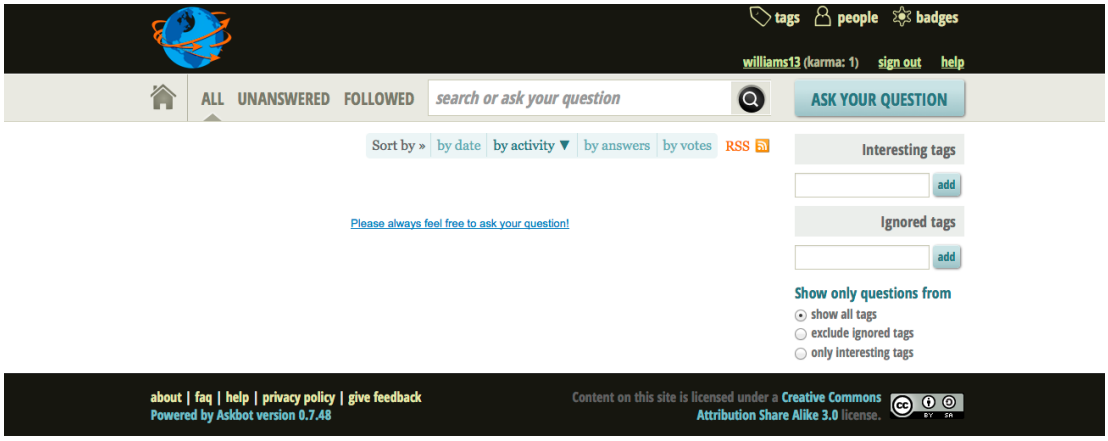



***Figure 17.*** *The new UV-CDAT Help Desk was developed to facilitate creating and asking questions regarding scientific questions and general use.*

## 3.8 Future Funding

Our project was designed to be a next-generation open-source tool for visualization and analysis of climate data that scales from small data sets on personal workstations through extreme-scale data residing on the supercomputers, as envisioned in DOE's ESM project. From the DOE perspective, our efforts will align closely with ESM to provide provenance and workflow functionality, diagnostics, and high-performance parallel analysis and visualization capabilities to the desktops of scientists who will apply these tools to inform decisions on meeting the energy needs of the nation. In addition, the team of developers will continue to collaborate with

national and international government agencies, universities, and corporations to extend parallel software capabilities to meet the challenging needs of ultra-scale, multi-model climate simulation and observation data archives.

# 4 Innovative Technology

## 4.1 Overview

UV-CDAT is an integrated framework that provides an end-to-end solution for data management, analysis, and visualization of ultra-scale data sets and is based on a client–server architecture. It builds on the following key technologies:

- The CDAT framework developed at LLNL for the analysis, visualization, and management of large-scale distributed climate data.
- ParaView, an open-source, multi-platform, parallel-capable visualization tool with recently added capabilities to better support specific needs of the climate science community.
- VisTrails, an open-source scientific workflow and provenance management system that supports data exploration and visualization.
- VisIt, an open source, parallel-capable, visual-data exploration and analysis tool that is capable of running on a diverse set of platforms, ranging from laptops to DOE's largest supercomputers.

These combined tools—along with others such as the R open-source statistical analysis and plotting software and custom packages (e.g., DV3D, ParCAT, and ViSUS)—form UV-CDAT and provide a synergistic approach to climate modeling, allowing researchers to advance scientific visualization of large-scale climate data sets. The UV-CDAT framework couples powerful software infrastructures through two primary means:

- Tightly coupled integration of the CDAT Core with the ParaView/VTK infrastructure to provide high-performance, parallel-streaming data analysis and visualization of massive climate-data sets. (Other tightly coupled tools include VCS, VisTrails, DV3D, and ESMF/ESMP.)
- Loosely coupled integration to provide the flexibility of using tools quickly in the infrastructure such as ViSUS, VisIt, R, and MatLab for data analysis and visualization as well as to apply customized data analysis applications within an integrated environment.

Within both paradigms, shown in Figure 18, UV-CDAT provides data-provenance capture and mechanisms to support data analysis via the VisTrails infrastructure.

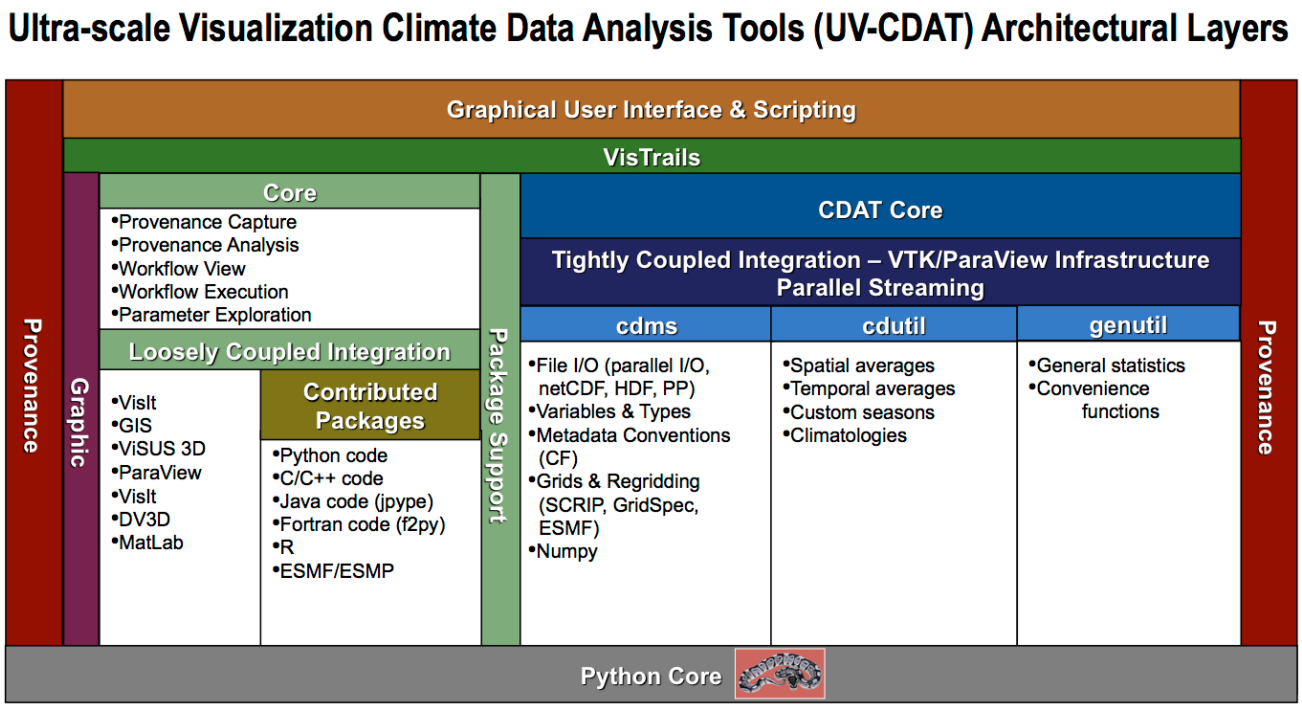



***Figure 18.*** *The UV-CDAT framework can accommodate both tightly coupled and loosely coupled model integration.*

Two active projects—the "*Ultrascale Visualization Climate Data Analysis Tools"* and the *"Visual Data Exploration and Analysis of Ultra-large Climate Data"*—delivered new never seen before capabilities to the climate science community under the UV-CDAT framework. Funded by the DOE Office of Science through its Earth System Modeling Program, these two projects share a joint vision for large-scale visualization and analysis for both observational and model-generated climate data, with the goal of delivering new capabilities into the hands of the climate scientists. The integrated software product, UV-CDAT, is a powerful and complete front end to a rich set of visual-data exploration and analysis capabilities well suited for climate-data analysis problems.

The UV-CDAT version 1.4 was released in September 2013 and is in production use. A UV-CDAT 2.0-beta release will be made available to the community for preview in January 2014. The UV-CDAT software source code is open source under the Berkeley Software Distribution license agreement.

## 4.2 Software Development

The UV-CDAT software followed best agile software development practices, which are a collection of software development methods based on iterative and incremental growth. Key points of agile software development include iterative development, continuous integration, working software, frequent releases, priority-based backlog, testing, regular scrums, and strategy (funding, goals, vision). Our software deployment process used some of the best-known methodologies and open-source tools.

This rigorous process involved robust testing and prototype use of new and established software libraries and packages before releasing them into the overall framework. Practical and technological experience with proven technologies of large community applications by the team, over the course of many years, allowed for careful integration and the build up of a completely new analytic software stack. For robustness, reliance on standards for data access, integration, processing, and visualization was key, as the standards constitute the basis for interoperability with other systems, clients, and server-side processes.

UV-CDAT followed these software practices from the beginning, by allowing a flexible architecture design to accommodate loosely coupled (i.e., fast) integration or tightly coupled (i.e., greater component interoperability)

integration. This design allowed the adoption of disparate application engines such as VisIt, ParaView, DV3D, Numpy, CDMS, and R while conforming to community standards such as netCDF, hierarchical data format (HDF), climate and forecast (CF) convention, and many others.

Over the short lifespan of the UV-CDAT project, the complexity of the software changed significantly. As mentioned earlier, all software components had to be directly relevant to the overall aim of UV-CDAT, and the design and implementation of the software at all levels complied with a useful programming practice technique all the way up to the large-scale software system. As the project's name suggests, the focus was on practice and experience with software for ultra-scale climate analysis and visualization tools. The UV-CDAT project covered and put into practice as many of these software engineering techniques as physically possible.

The key steps to be completed before submitting new software to the UV-CDAT software stack were design, implementation, and the all-important team testing. No exceptions were made. Software processing steps included:

- Providing detailed design of software component sub-system and the niche that it would serve in the overall framework. Although there are many redundancies in the framework, this process helped to prioritize the work and avoid system overlap.
- Presenting design and programming implementation to the group during teleconferences and face-to-face meetings.
- Documenting sub-component (i.e., library or package) and tools that aid in solving software construction problems.
- Explaining methods/techniques to cope with the special demands of the software component.

For any reported bug submitted or feature requested, a developer creates a new topic from the most stable source code branch, which is also referred as the *master*. Then the developer makes minor or major changes to the code as required by the bug or feature requested and registers each change set as a commit object. When ready, the developer publishes their code to the next branch, which contains the most current integrated code. As soon as that happens, the UV-CDAT testing machines detect this change and perform a continuous build of the next code branch. Once finished, each of the testing machines run tests and submit its complete report to the dashboard site, as shown in Figure 19.


My CDash All Dashboards Log Out — Thursday, February 07 2013 23:12:19 EST

UV-CDAT

Dashboard Calendar Previous Current Next Project Settings

No file changed as of **Wednesday, February 06 2013 - 20:00 EST** — Show Filters Advanced View Auto-refresh Help

**Nightly**

| Site | Build Name | Update | Configure | | Build | | Test | | | Build Time | Labels |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Files | Error | Warn | Error | Warn | Not Run | Fail | Pass | | |
| discover29 | UVCDAT-SLES11_gcc471-master | 0 | 0 | 0 | 1 | 1 +1 -1 | 0 | 42 | 0 | 21 hours ago | (none) |

**Nightly-next**

| Site | Build Name | Update | Configure | | Build | | Test | | | Build Time | Labels |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Files | Error | Warn | Error | Warn | Not Run | Fail | Pass | | |
| kargad-linuxvm-cdat | UVCDAT-kargad_linux-next | | | | | | | | | Expected build | (none) |
| meryem.llnl.gov | UVCDAT-meryem_Mac_10.6-next | 0 | 0 | 0 | 1 +1 | 0 | 0 | 42 +42 | 0 | Feb 06, 2013 - 21:33 EST | (none) |
| pcmdi11.llnl.gov | UVCDAT-pcmdi11_RedHat_6-next | 0 | 0 | 0 | 1 +1 | 1 +1 -2 | 0 | 6 +1 | 36 -1 | 22 hours ago | (none) |
| placid.lanl.gov | UVCDAT-RHEL6.3-x86_64-next | 0 | 0 | 0 | 0 | 1 +1 -1 | 0 | 1 | 39 | 18 hours ago | (none) |
| vishnu | UVCDAT-arch_gcc470-next | 0 | 0 | 0 | 0 | 2 +1 -1 | 0 | 0 | 42 | 5 hours ago | (none) |
| kargad.kitwarein.com | UVCDAT-kargad_maclion-next | 0 | 0 | 0 | 0 | 2 +1 -1 | 0 | 0 | 42 | 20 hours ago | (none) |
| samus.lanl.gov | UVCDAT-samus_ubuntu_12.04_x86_64_gcc463-next | 0 | 0 | 0 | 0 | 2 +1 -1 | 0 | 0 | 42 | Feb 06, 2013 - 23:00 EST | (none) |
| yavin | UVCDAT-yavin_ubuntu_64bit_gcc463-next | 0 | 0 | 0 | 0 | 2 +1 -1 | 0 | 0 | 42 | 18 hours ago | (none) |

**Nightly-master**

| Site | Build Name | Update | Configure | | Build | | Test | | | Build Time | Labels |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Files | Error | Warn | Error | Warn | Not Run | Fail | Pass | | |
| placid.lanl.gov | UVCDAT-RHEL6.3-x86_64-master | | | | | | | | | Expected build | (none) |
| kargad-linuxvm-cdat | UVCDAT-kargad_linux-master | | | | | | | | | Expected build | (none) |
| meryem.llnl.gov | UVCDAT-meryem_Mac_10.6-master | 0 | 0 | 0 | 1 +1 -1 | 0 | 0 | 42 | 0 | Feb 06, 2013 - 20:24 EST | (none) |
| pcmdi11.llnl.gov | UVCDAT-pcmdi11_RedHat_6-master | 0 | 0 | 0 | 0 -1 | 2 +2 -1 | 0 | 5 -1 | 37 +1 | 17 hours ago | (none) |
| kargad.kitwarein.com | UVCDAT-kargad_maclion-master | 0 | 0 | 0 | 0 | 2 +1 -1 | 0 | 3 | 39 | 17 hours ago | (none) |
| vishnu | UVCDAT-arch_gcc470-master | 0 | 0 | 0 | 0 | 2 +1 -1 | 0 | 0 | 42 | 22 hours ago | (none) |
| samus.lanl.gov | UVCDAT-samus_ubuntu_12.04_x86_64_gcc463-master | 0 | 0 | 0 | 0 | 2 +1 -1 | 0 | 0 | 42 | Feb 06, 2013 - 22:00 EST | (none) |
| yavin | UVCDAT-yavin_ubuntu_64bit_gcc463-master | 0 | 0 | 0 | 0 | 2 +1 -1 | 0 | 0 | 42 | 20 hours ago | (none) |

***Figure 19**. The UV-CDAT dashboard on a particular day. Errors during configure, build, or run time are shown in red with warnings in green.*

Based on the quality of the code, and the regression test results, a gate-keeper (maintainer of the UV-CDAT repository) could choose to approve, deny, or request improvements on the submitted branch. If approved, the change is published (merged) in the UV-CDAT master branch. Advantages to this software process include:

- Efficient management of contributions from a large number of developers located in different geographical locations.
- Support for continuous development. By utilizing distributed version control (Git) and topic-based workflow, developers can work in parallel.
- Support for test-driven development. Each branch is tested continuously and nightly. Each new feature branch is encouraged and accompanied by a test for that feature.

### 4.2.1 Test-Driven Development (Kitware)

For the complete test-driven agile software process, the UV-CDAT team adopted and extended the model used by Kitware. For UV-CDAT's development of large libraries and packages, Kitware implemented its novel software process based on methods from agile development and test-driven development. The process incorporates source code control, using Git source code repository; build management, using CMake; and regression testing, using CTest and CDash. A highlight of this process is that it runs continuously and simultaneously on multiple platforms, and produces its results on web-accessible dashboards. Thus, as software is checked into the source code repository, it is immediately tested and results posted. Developers rapidly obtain feedback on their changes and correct problems immediately. This process, along with user access to the code repository, is demonstrated in Figure 20.

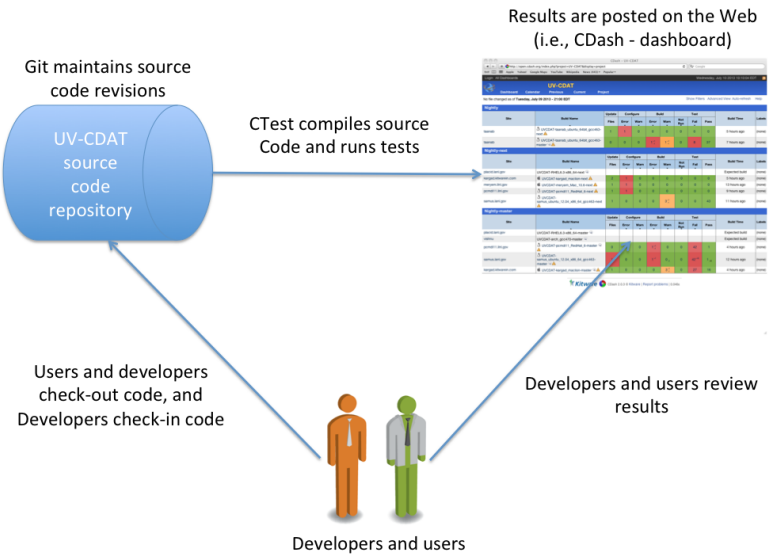



***Figure 20.*** UV-CDAT's Git source code revisions and test-driven agile software process.

### 4.2.2 Code Repository (Git and Github)

UV-CDAT developers are located throughout the United States at DOE facilities, universities, private companies and internationally. To enable them to work together on this large-scale analysis and visualization software, we provide the source code in an open-source Internet-accessible revision control system called Git. With Git, UV-CDAT developers have write access to the software repositories, enabling them to make changes to the source code. Everyone has read access to the repositories, so one could download the most up-to-date stable or

development version of the software. A stable release of the source code can be downloaded from a separate branch of the distribution directory found at the following URL http://uv-cdat.org/. The UV-CDAT Git developer's repository can only be accessed from the bleeding-edge repository for development effort. There are several ways to access the UV-CDAT Git repository:

- GitHub Web Access: To browse around or download a few individual files or submit and check out bug fixes, the best tool is the web-based GitHub interface.
- Anonymous Git: To access the Git repository anonymously, a Git client is required to access the entire UV-CDAT repository or choose the module needed and check it out. For example, to get only the VisTrails module, use: "git clone git://vistrails.org/vistrails.git"
- Developer Git Access: HTTPS is used as basic authentication for logging in to the Git repository, which requires a user name and password.

For more help on using Git, consult the Git website or Git book [Chacon, 2013]. The website provides a list of clients and useful links.

### 4.2.3 Software Installation (CMake)


As a first step toward introducing an agile software development process to UV-CDAT, we converted its build process to use CMake. This conversion allowed us to use CTest and CDash to test and monitor the development process. CMake is a family of open-source tools designed to build, test, and package software. In the last few years CMake has been adopted by many popular open source projects such as KDE, OpenSceneGraph, and Qt5. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of the developer's choice. CMake is quite sophisticated: it is possible to support complex environments requiring system introspection, pre-processor generation, code generation, template instantiation, and cross-compilation. UV-CDAT leverages CMake to enable users to build and package UV-CDAT on personal and supercomputers. Using the CMake GUI or command line interface, developers could configure most of the packages to fit their need and requirement. Once configured, CMake generates platform specific makefiles (Ninja or GNU). On Unix, Linux, and Mac OSX systems, developers then could run CMake with parallel jobs supported by the OS to build UV-CDAT in parallel. All of the packages of UV-CDAT are installed at a common location, which then can be bundled to create platform specific installers.

UV-CDAT employs CMake 2.8's ExternalProject module to build each of its approximately seventy dependencies, libraries and packages. The ExternalProject module makes it easier to build external software components on which the project depends. An "external project" is one for which the source code is available but is not necessarily part of the main build process. The ExternalProject_Add function makes it possible to say "download this project from the internet, run its configure step, build it and install it" in just a few lines of code in the CMakeLists.txt file. All of the time-intensive processing that occurs for each step is deferred until build time.

The basic concept of ExternalProject is simple: given an external source of software (URL to a uv-cdat.tar.gz file, Git repository, or local directory), execute the sequence of commands necessary to build and install that software such that it can be referred to (include, link, run) from the main project.

By design, ExternalProject is modular and compact. With this we can easily extend UV-CDAT to build additional packages when the need arises. Additionally, just employing CMake as the build system of UV-CDAT allows us to very easily extend and customize its build process in a variety of ways. We can use CMake to give the users numerous build options, allowing a developer to easily build portions of UV-CDAT as needed. This allows us to achieve a requirement goal of distributing a UV-CDAT-lite (e.g., only building CDAT or components of CDAT such as CDMS).

Figure 21 shows the build order as computed by the build system for Python and related packages. Each package has at most three files that define its version information, location of source code, dependencies, and exact command to build the package, e.g., python_pkg, python_deps, and python_externral. The build system enables users to download packages via Git, HTTP, or HTTPS protocol. Also, if a user expects no Internet access, she can download the packages for later use. The build system, shown in Figure 21, provides flexibility to use a particular system package if needed and also supports parallel build.

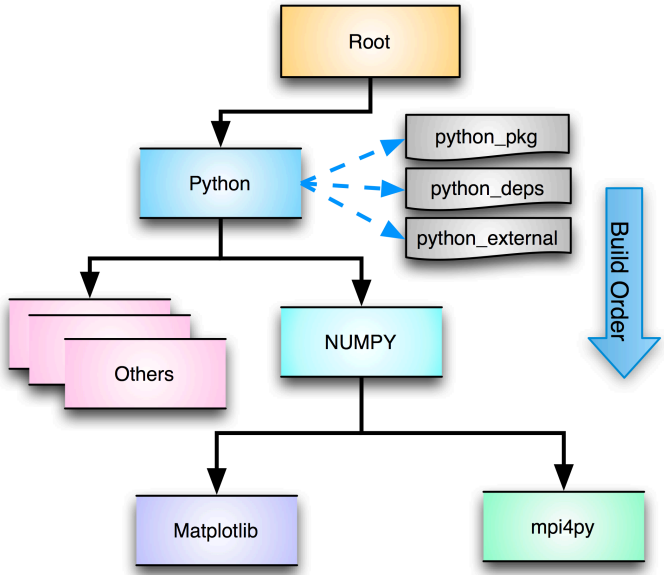



***Figure 20.*** *A cross-platform build system that generates native makefiles and workspaces.*

For more information on CMake and its documentation, see:
http://cmake.org/cmake/help/documentation.html.

### 4.2.4 Software Test Suite (CTest)

CTest is a testing tool distributed as a part of CMake. It can be used to automate updating (using Git for example), configuring, building, testing, performing memory checking, performing coverage, and submitting results to CDash dashboard system. There are two basic modes of operation for CTest. In the first mode, CMake is used to configure and build a project, using special commands in the CMakeLists.txt file to create tests. CTest can then be used to execute the tests, and optionally upload their results to a dashboard server. This is what is handled in this tutorial. In the second mode, CTest runs a script (using the same syntax as CMakeLists.txt) to control the whole process of checking out/updating source code, configuring and building the project, and running the tests. Including *enable_testing()* enables all the necessary resources required for the testing. A test can be added to UV-CDAT by utilizing *add_test* command provided by CMake. Once the project has been built, we can execute all of the tests simply by providing *test* argument to make.

### 4.2.5 Nightly Monitoring (CDash)

The next step not only executes the tests but also logs their results in such a way that they can be reviewed easily. The result of a test run, reformatted for easy review, is called a "dashboard." A dashboard can be submitted to a central server, such as CDash, which is described at www.cdash.org and my.cdash.org.

CDash is an open-source, web-based software-testing server. CDash aggregates, analyzes and displays the results of software testing processes submitted from clients located around the world. CDash provides three types of dashboard submissions:

- *Experimental* means the current state of the project. An experimental submission can be performed at any time, usually interactively from the current working copy of a developer.
- *Nightly* is similar to experimental, except that the source tree will be set to the state it was in at a specific night time. This ensures that all "nightly" submissions correspond to the state of the project at the same point in time. "Nightly" builds are usually done automatically at a preset time of day.
- *Continuous* means that the source tree is updated to the latest revision, and a build/test cycle is performed only if any files were actually updated. Like "Nightly" builds, "Continuous" ones are usually done automatically and repeatedly in intervals.

UV-CDAT enables the creation and submission of dashboards by utilizing the special include from CMake and providing a CTestConfig.cmake file to describe dashboards site and various preferences for CDash submissions. UV-CDAT provides scripts for setting up the dashboards on any computer that can build UV-CDAT. UV-CDAT dashboards submit continuous and nightly reports to the UV-CDAT CDash site (http://open.cdash.org/index.php?project=UV-CDAT&date=2013-08-15&display=project).

## 4.3 Data Files

Climate data files are either raw data—from simulation runs and observational information—or metadata—information describing the raw data, what is to be done with it, and how to use it. Formats on how to store and retrieve multidimensional data and describing metadata are many. In the climate modeling community, the many types of formats include the netCDF [UCAR, 2008], HDF [WMO, 2009], the GRIdded Binary (GRIB) [HDF Group, 2009], and the Post Processing (PP) format [PP, 2013], to name only a few.

However, in recent years, more groups running climate model simulation codes are opting to store their data in the netCDF format. NetCDF has evolved over time and has merged efforts with HDF for wider use in the creation, access, and sharing of geoscience (i.e., simulation and observational) data. That is, the netCDF API now sits on top of the HDF software stack as release version 4.x and takes full advantage of advanced HDF5 features such as grouping, compression, and chunking. For community archives, such as those held in the ESGF distributed federation data enterprise system, data are mainly stored in the netCDF format.

The climate modeling and observational communities have also selected a de facto methodology standard for defining the metadata that provides definitive descriptions of what the data in each variable represents and the spatial and temporal properties of the data. Combined with the netCDF API, the Climate and Forecast (CF) [CF, 2009] metadata convention enables users of data from different sources to decide which quantities are comparable, and facilitates building applications (such as UV-CDAT) with powerful extraction, regridding, and display capabilities. This metadata convention makes it possible for other geo-referencing data such as atmospheric, ocean, atmospheric chemistry, biogeochemistry, and satellite observations to be compared and displayed together with little or no effort on the part of scientists.

To ensure climate modeling and data centers can easily produce the netCDF CF metadata convention output, LLNL has developed the Climate Model Output Rewriter (CMOR) [CMOR, 2009]. The structure of the files created by CMOR and the metadata they contain fulfills the requirements of many of the climate community's standard model experiments [Taylor, 2011]. A new version of CMOR was developed for Climate Model Intercomparison Project Phase 5 (CMIP5) model assessments with new grid storage capabilities and additional CF functionalities for observational and multi-model comparisons. As a component of UV-CDAT, CMOR has a built in checker to make sure the model output complies with the netCDF CF conventions. This checker guarantees the quality of the output of the model data before it is archived into the ESGF CMIP5 distributed data repository.

Implemented as part of UV-CDAT, the CDMS [Drach, 2007] is used to automatically locate and extract metadata (i.e., variables, dimensions, grids, etc.) from the multi-model collection of model runs and analysis files. CDMS is defined as an object-oriented data management system, specialized in organizing multidimensional, gridded data

used in climate analysis and simulation. As a fully CF-compliant data-access tool, UV-CDAT (via CDMS) allows users to seamlessly read data from multiple sources for intercomparison multi-model, observational, and reanalysis studies. In addition to reading in netCDF CF-compliant data, UV-CDAT can also write data in this format and convention. UV-CDAT can also read in HDF, GRIB, PP, binary, and other popular climate data file formats.

UV-CDAT expedites the manipulation of data arrays with missing values, an often-troublesome issue in geophysical applications. This problem is resolved by extending the Numerical Python ("Numpy") masked-array (MA) function to a masked-variable (MV) construct that retains information on the metadata attributes of data arrays. It also allows an extensive collection of Numpy mathematical operations to be executed. (Figure 22.)

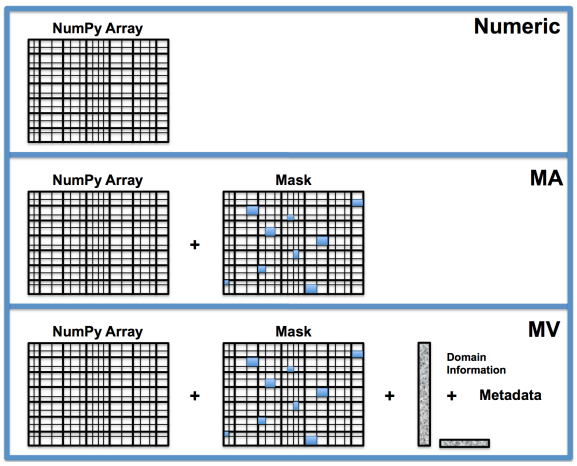



***Figure 21.*** *UV-CDAT supports the combination of many types of array objects such as the Python Numpy Array, where all elements in the multidimensional array have the same data type (real, integer, etc.); the Masked Array (MA), a Numpy array with an optional missing data mask; and the Masked Variable (MV) (or Transient Variable), a MA with domain and metadata.*

### 4.3.1 Analysis

Exploratory visualization is an important aid in the analysis process for climate scientists. Using UV-CDAT, they can perform different types of analyses, including intra-model comparison by focusing on different variables from the same model, inter-model comparison by focusing on the same variable from multiple models, and verification and validation against observational data.

While UV-CDAT provides a drag-and-drop interface, making it convenient for climate scientists to create and customize visualizations, there is also some scope for improving the quality of the visualizations to better cater to the analytical needs of scientists. Concise visual representation provides a transparent window to the data, and by interacting with it, scientists can derive new insights about simulations, model input and output, model structure, etc. On the other hand, erroneous visualization design or high complexity of visual representations can impede the analysis process by causing expensive visual search for patterns.

Our goal was to eliminate the probability of such erroneous designs as much as possible. To meet this goal, in tight collaboration with a pool of climate scientists, we conducted an exploratory study of climate data visualizations. The study analyzed a large set of static climate data visualizations for identifying and categorizing their potential shortcomings in terms of visualization design. The outcome from this study, which is still under way, will be a set of best practices for visualization design to serve as model visualizations for specific analysis scenarios. They will guide climate scientists in making informed choices about matching their intended tasks with optimal visual representations.

### 4.3.2 Visualizations

UV-CDAT provides support for different visualizations for different analysis scenarios. By integrating scientific visualization methods like volume rendering with information visualization techniques using coordinated multiple views, UV-CDAT helps capture salient patterns in the data from various perspectives.

Visualizing spatial correlation and variance is one of the main tasks in climate data analysis, and can be accomplished using the various 3D volume visualizations and geographical maps. Spatial aggregation methods with region masks are integrated within UV-CDAT and can be applied directly on the selected visualization types.

Another important task is detecting temporal trends and anomalies. This is supported within UV-CDAT in two ways: encoding time along the third dimension, where temporal patterns are represented by color maps, and through coordination and linking between maps and line charts. By slicing through the maps and linking the different spatial positions to time series in line charts, scientists can observe the temporal behavior of variables at those locations**.**

One analysis task that is underrepresented within the UV-CDAT visualization framework is analyzing multidimensional data. This is important because scientists want to compare the behavior of multiple output variables, or the components of multidimensional output variables like Gross Primary Productivity (GPP) at different spatial locations. We are currently working to add visualizations that support multidimensional analysis, such as parallel coordinates, heat maps, etc.

### 4.3.3 Scripting

Scientists will always have pressing needs to write programs to ingest, manipulate, and display data and repeat these very same tasks as new data becomes available. Therefore, we sought to minimize this effort as much as possible so that scientists can focus on research. To this end, UV-CDAT makes use of an open-source, object-oriented, easy-to-learn scripting language (Python) to link together disparate software subsystems and packages to form an integrated environment for climate data analysis. The resulting software environment is user-friendly, reusable, portable, and promotes the sharing of common standards within the community. In this environment, specialized data access, diagnostic algorithms, and display software cooperate under a variety of user interfaces including command line, stand-alone application scripts, GUIs, and web browsers.

Besides being able to enter interactive command line calls at the Python prompt, users can produce a complete Python script (or series of commands) and save it to a file. This script can be executed any time, reopened and modified, and/or shared with colleagues. Scripts save the user from doing repetitive actives by describing exactly which commands to execute and in what sequence, so that the operations on the data are doing the right thing and in the correct sequence. Python is widely used in other scientific application areas, and there are many books, tutorials, and websites on this popular fourth-generation language.

### 4.3.4 Workflow and Provenance Control

Underlying the UV-CDAT GUI is an infrastructure to support the definition and execution of analysis and visualization tasks as well as track the provenance of data and results. Scientific workflows serve to fully encode the specification of the tasks in an intuitive dataflow structure. In addition, it is possible to uniformly capture provenance information during workflow execution. By utilizing the VisTrails infrastructure, UV-CDAT is able to construct and execute scientific workflows and capture the provenance of data, results, and workflow development. The UV-CDAT GUI provides a high-level interface for constructing and manipulating workflows, but users can directly view and modify them if desired. See also Section 4.12 for more information on workflow and provenance.

#### *4.3.4.1 Workflow*

Workflow systems provide well-defined languages for differentiating complex tasks from simpler ones. They also capture complex processes at various levels of detail and systematically record the provenance information necessary for automation, reproducibility, and sharing of results.

Scientific workflows are often used to perform data-intensive tasks and are represented as dataflow networks where the execution order is determined by the flow of data through the workflow. Scientific workflow systems offer a number of advantages over programs and scripts for constructing and managing computational tasks. In addition to providing a simple programming model, many systems provide intuitive visual programming interfaces, which make them more usable for users who do not have substantial programming expertise. The structure in workflow definitions also enables a series of operations and queries that simplify the manipulation and re-use of workflows. From now on in this document, we will use the terms scientific workflow, workflow, pipeline, and dataflow interchangeably.

Formally, a workflow specification is defined by sets of modules, input ports, output ports, connections, and a function that assigns a unique module to input and output ports. Additionally each port (input or output) has an associated type and a name that is unique across the ports of the same module. A module signature is defined as a set of pairs, where each pair contains a port name and its type. In addition, each module has a set of parameters. Each parameter has a unique name across the set of parameters of the same module.

A workflow instance is a specification combined with bindings that provide values to parameters in the modules. A workflow run is the execution of a pipeline in the order determined by the network of modules and connections.

#### *4.3.4.2 Provenance*

In the context of scientific workflows, data provenance is a record of the derivation of a set of results. There are two distinct forms of provenance: prospective and retrospective. Prospective provenance captures the specification of a computational task (i.e., a workflow specification or instance); it corresponds to the steps that need to be followed (or a recipe) to generate a data product or class of data products. Retrospective provenance captures the steps that were executed (i.e., the workflow run) as well as information about the execution environment used to derive a specific data product—a detailed log of the execution of a computational task.

Causality is also important to provenance. Causality is the collection of dependency relationships between data products and the processes that generated them. Causality can be inferred from both prospective and retrospective provenance. Data provenance also includes user-defined information, such as documentation that cannot be automatically captured but records important decisions and notes. This data is often captured in the form of annotations.

### *4.3.5 Parallelism*

UV-CDAT provides several methods for performing parallel tasks, including data parallelism, task parallelism, and hybrid data/temporal parallelism. The methods available include interactive and non-interactive visualization and analysis. Tools and features for accessing parallel methods include ParaView, VisIt, ParCat, and Distributed Arrays. Large-scale data parallelism using ParaView or VisIt is enabled by a client–server architecture. A server is instantiated on a parallel-capable machine, and the client connects to the server and communicates instructions to the server. For interactive sessions, the UV-CDAT GUI acts as a client. Users can connect to the server by providing connection information in the GUI or the interactive Python shell provided by the GUI. Once the server has finished loading the data and processing it, the GUI is updated with the results from the parallel job. Non-interactive batch processing is also available for parallel processing. UV-CDAT supports scripting in Python and calls parallel methods already compiled in C/C++/Fortran. Hybrid spatial/temporal parallelism is also available in ParaView by using the spatio-temporal pipeline in a batch-processing mode. The spatio-temporal pipeline can greatly increase parallel performance for data sets with large spatial and temporal resolutions. VisIt offers additional support for applying R algorithms in parallel to spatially partitioned data pieces. ParCAT delivers temporal parallelism capability (in batch) for high temporal resolution and smaller spatial extents data sets.

Distributed arrays are a feature in UV-CDAT, which allows the computation and processing of large-scale arrays that would normally not fit on one node of memory.

UV-CDAT provides a number of innovative mechanisms for exploiting high-performance computing resources to analyze and visualize data sets using distributed memory parallelism. Parallel visualization within UV-CDAT is supported through VisIt and ParaView. Data reduction of large-scale data sets to generate summary statistics and climatologies is supported through ParCAT. Temporal-spatial parallelization is supported through both ParaView and ParCAT. Distributed array-based parallelism is supported through MPI-2 one-sided operations coupled with a global address space paradigm. These complementary parallel analysis and visualization mechanisms cover a wide range of common use cases in ultra-scale climate data processing. These techniques are described in greater detail in Sections 4.6, 4.7, and 4.13.

### *4.3.6 Supported Operating Systems on Personal Workstations to Supercomputers*

UV-CDAT is designed to support a variety of computational platforms spanning a climate scientist's Mac OS laptop, to Linux-based workstations, to large-scale high-performance systems such as Titan at ORNL. Binary distributions of UV-CDAT are made available to the climate science community through the UV-CDAT website and include a variety of supported operating systems including Mac OS, Fedora, Ubuntu, Redhat, and CentOS. Future support for the Windows platform is in the works. In addition to binary distributions, UV-CDAT is also available as source. UV-CDAT is currently available at major computing facilities including systems at the Oak Ridge Leadership Computing and the National Energy Research Scientific Computing Center.

## 4.4 Access to ESGF's Distributed Archive

The UV-CDAT–ESGF connection provides a user-friendly interface, shown in Figure 23, for search and retrieval of any data sets stored in any of the data nodes on ESGF. A user can search the data sets by selecting the facets through the search categories and/or entering search terms in the text field. For facet selection, a user can select multiple facets of the same category simultaneously. Once a facet is selected, all other categories located below the selected category are updated instantly. Any facets or search terms selected for the search are automatically added to the list of current selections in the current selection panel. The current selection panel has a text box that allows the users to highlight an item shown in the current selections, and clicking on the remove button removes the highlighted selection and triggers an instant update of facets. In addition to the remove button, the current selection panel also provides toggle and reset buttons, which function to toggle the current selection panel and to reset the list of current selections back to its default setting. By default, no selections are made and no replicas are searched. However, the user has the option to show all replicas by selecting the "Show All Replicas" checkbox under the data node selection panel. The data node selection panel, which is next to the current selection panel, provides the user with the option to select the data nodes from which the search engine will search through its data sets. By default, all data nodes are selected but not highlighted, which means that all data nodes are searched if no data node selections are made. In the data nodes selection panel, two buttons are provided: to toggle the data node selection panel and to select all the data nodes listed in the data node selection panel. Once the user clicks the "Search" button, the search engine takes the list of current selections and searches through all the selected data nodes to retrieve the relevant data sets shown in the search result panel.

For each search result, a total number of files available are displayed on top of the search result panel. The search results are divided into multiple pages for display. Each page has a maximum of 500 files by default, but can change file retrieval limits to the ESGF tab in the UV-CDAT preference window. A user may navigate to a particular result page by entering a page number or clicking on the "Next" button to go to the next page. For each of the results page, all the search results are displayed in a tree format. The user can choose to delete any of the selected items from the result tree or click on "Delete All" to delete all the results. There is also a "Collapse All" button to collapse all the results in the result tree. Inside the result tree, the user has the option to download the selected file via GridFTP, HTTPServer or OPENDAP. However, GridFTP has not been implemented yet. In order

to access any of the data, the user must be authenticated and authorized. The user can enter her credential by clicking on the key image above the text search.

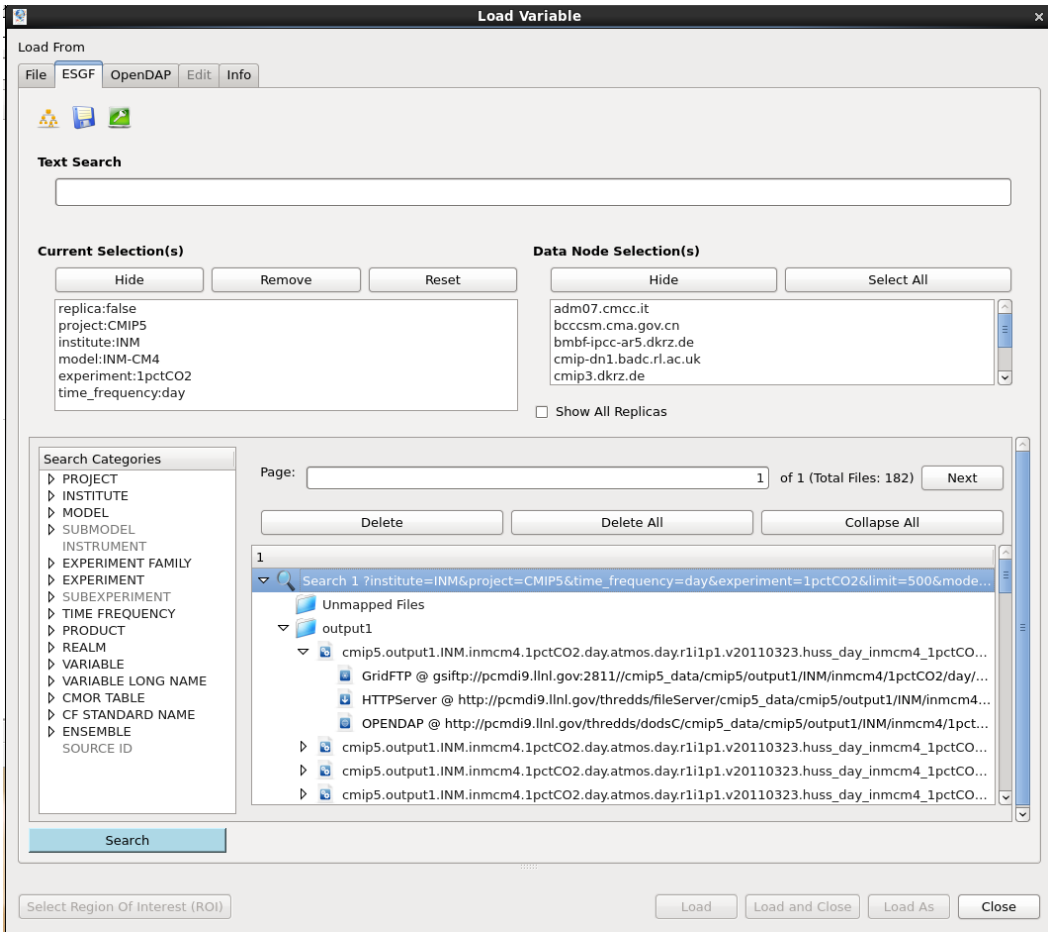



***Figure 22.*** *UV-CDAT's ESGF interface to access secure distributed data.*

## 4.5 Climate Analysis Tools

### *4.5.1 Cdutil*

Cdutil is an LLNL-developed package that retains metadata information after some sort of data manipulation. It is geared toward climate-specific applications such as time extraction, seasonal average, bounds setting, vertical interpolation, variable massager (i.e., prepping variables in order to compare them, such as masking/regridding), region extraction, and other similar functions.

### *4.5.2 Genutil*

Genutil is a package developed by LLNL to facilitate day-to-day climate analysis and diagnosis in UV-CDAT. These tools are not climate specific, but more "array" specific. (See Figure 22.) They are "metadata-smart," retaining metadata information after some sort of data manipulation (a UV-CDAT trademark). Example of Genutil tools include statistics, array growing (to expand a data array before comparing data with a different number of dimensions, (e.g., applying a 2D land/sea mask to a 3D data set), color manipulation by name, status bars, string templates, and selecting non-contiguous values across a dimension.

### 4.5.3 ESMF/ESMP

The ESMF is software for building coupled weather, climate, and related models. UV-CDAT tightly integrates ESMF's linear, quadratic, and conservative regridding methods through a thin Python layer (ESMP). The regridding methods are also bundled in the netCDF Climate Forecast library (LibCF). Developed at UCAR and Tech-X, it makes it easier to read and write data conforming to the Climate and Forecast conventions. With LibCF, UV-CDAT gains access to data stored in the GRIDSPEC gridded layout (see Figure 24). UV-CDAT also tightly integrates the multi-dimensional linear interpolation methods of LibCF, which can be used as an alternative to ESMF interpolation.

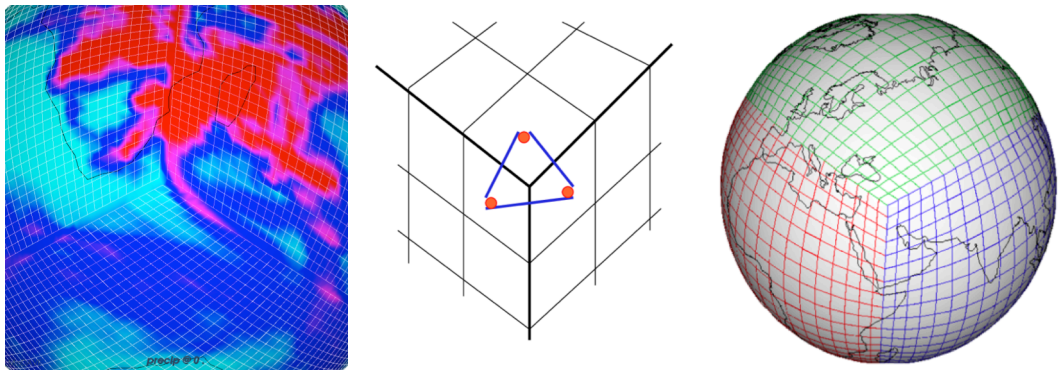


***Figure 23.*** *These images show cell-centered precipitation on the cubed-sphere and how the connectivity between cubed-sphere tiles is used in order to fill in the color of the inter-tile gaps and of the triangular cell covering the three-tile corner. This was done explicitly by showing the fields on each tile, and creating 12 gap and 8 corner cell grids.*

## 4.6 Visualizations Tools

### 4.6.1 DV3D

DV3D is a VisTrails package of high-level modules for UV-CDAT providing user-friendly workflow interfaces for advanced visualization and analysis of climate data at a level appropriate for scientists (Figure 25). It builds on VTK, an open-source, object-oriented library, for visualization and analysis. DV3D provides the high-level interfaces, tools, and application integrations required to make the analysis and visualization power of VTK readily accessible to users without exposing details such as actors, cameras, renderers, and transfer functions. It can run as a desktop application or distributed over a set of nodes for hyperwall, as shown in Figure 5, or distributed visualization applications.

The DV3D package offers scientists a set of coordinated, interactive 3D views (i.e., plots) of their data sets, as shown in Figure 25. Each DV3D plot type offers a unique perspective by highlighting particular features of the data. Multiple plots can be combined synergistically (within a single cell or across multiple cells) to facilitate understanding of the natural processes underlying the data. For example, the plot types include:

- The *Slicer* plot provides a set of slice planes that can be interactively dragged over the data set. A slice through the data volume at the plane's location is displayed as a pseudocolor image on the plane. A slice through a second data volume can also be overlain as a contour map over the first. This tool allows scientists to quickly and easily browse the 3D structure of the data set, compare variables in 3D, and probe data values.
- The *Volume* render plot maps variable values within a data volume to opacity and color. It enables scientists to create an overview of the topology of the data, revealing complex 3D structures at a glance. Due to the complexity of creating useful transfer functions, the art of generating volume renderings has in the past been relegated to visualization professionals. DV3D offers interfaces that greatly simplify this process, enabling interactive volume rendering to play an important role in the scientist's data exploration process.

- The *Isosurface* plot displays an isosurface derived from one variable's data volume and colored by the spatially correspondent values from a second variable's data volume. It can produce views similar to a volume rendering while facilitating the comparison of two variables.
- The *Hovmoller* slicer and volume render plots are similar to the 3D slicer and volume render plots described above except that they operate on a data volume structured with time (instead of height or pressure level) as the vertical dimension. This plot allows scientists to quickly and easily browse the 3D structure of spatial time series.
- The *Vector* slicer plot provides a set of slice planes that can be interactively dragged over a vector field data set. A slice through the field at the plane's location is displayed as a vector glyph or streamline plot on the plane. This plot allows scientists to browse the structure of variables (such as wind velocity) that have both magnitude and direction.

All configuration operations are saved as VisTrails provenance. The provenance trail contains a record of all workflow construction and configuration operations that contributed to the current visualization, making it easy to revert to an earlier configuration of the workflow at any stage of development.

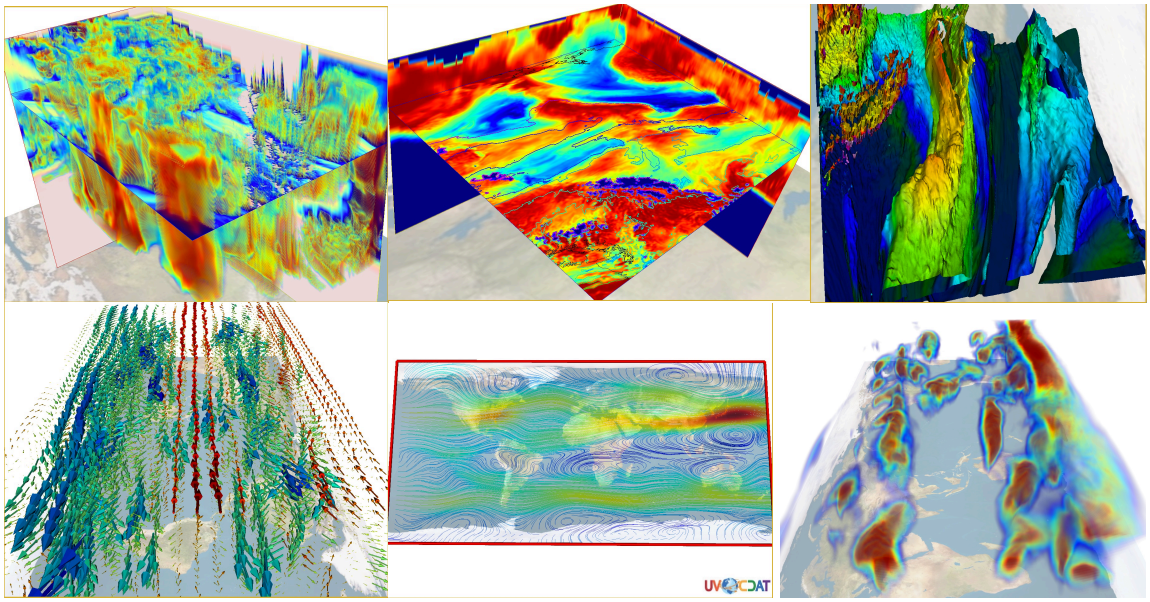


***Figure 24.*** *A selection of DV3D plots: Volume render withsSlicer (upper left), volume slicer (upper middle),isosurfaces (upper right), glyphvVolume of 3DvVectors (lower left), streamline slicer (lower middle), andvVolume render (lower right).*

All of these plot types offer the following features:

- Animating over one of the data dimensions (typically time) provides a very effective method for viewing and browsing 4D data.
- The VisTrails workflow interface provides individual GUIs for each module and a powerful customization for developing visualization and analysis pipelines.
- A rich selection of interactive query, browse, navigation, and configuration options facilitates exploratory visualization.
- Integration with the VisTrails spreadsheet provides multiple synchronized plots for desktop or hyperwall (see Figure 5).
- Integration with the VisTrails provenance architecture provides transparent collection and comprehensive management of workflow and data provenance.

- The underlying VTK architecture provides active and passive 3D stereo visualization support.
- Seamless integration with CDAT's CDMS and other climate data analysis tools provides extensive climate data processing and analysis functionality.

### 4.6.2 Matplotlib

Matplotlib is a generic Python 2D and 3D plotting library whose primary data type is Numpy. Since UV-CDAT's primary datatype, CDMS, is based on Numpy, Matplotlib naturally lends itself to be incorporated into UV-CDAT. While only one predefined Matplotlib plot, the histogram, is selectable in the UV-CDAT GUI, users familiar with Matplotlib can easily edit the VisTrail's Matplotlib module at runtime to produce any Matplotlib visualization desired. Figure 26 shows two custom Matplotlib plot types.

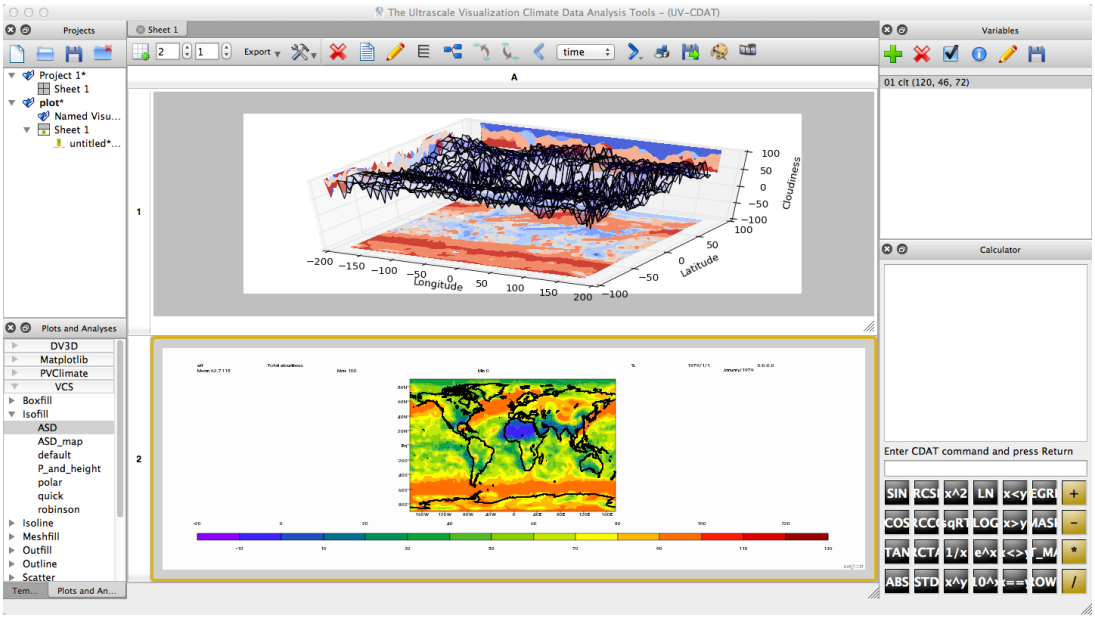



***Figure 25.*** *Matplotlib can be customized and added to UV-CDAT. Shown here is a 3D Matplotlib plot type at the top and a 2D CDAT beneath. Efforts are underway to have more Matplotlib plot types directly in UV-CDAT.*

### 4.6.3 ParaView

The integration of ParaView into the UV-CDAT framework has brought a wealth of analysis and visualization capabilities to the framework. The serial abilities of ParaView have been tightly integrated into UV-CDAT, which brings with it several benefits. For example, a person using a set workflow can also access the parallel capabilities of ParaView. Several steps have also been taken to tightly integrate ParaView with VisTrails. Several ParaView-specific VisTrails modules have been created, including PVContourRepresentation and PVGenericCell, which support one or more representations for its input.

In response to the demands and use cases of the UV-CDAT program, a new parallel pipeline helper has been developed to take advantage of temporal parallelism, called the UV-CDAT spatio-temporal pipeline. With ParaView and VisTrails so tightly integrated, when ParaView plots are used in UV-CDAT, VisTrails automatically collects provenance information about the plots, which can then be shown in the pipeline viewer. Figure 27 illustrates a ParaView plot in UV-CDAT, and the corresponding VisTrails pipeline workflow for that plot.

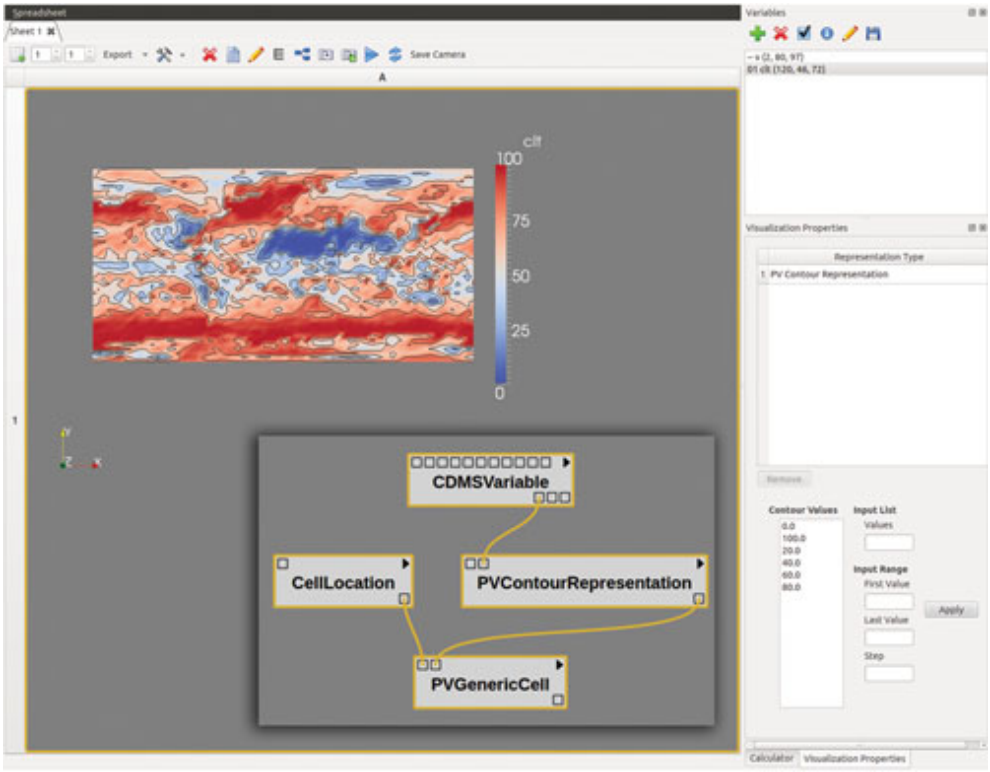



***Figure 26.*** *ParaView plot and corresponding VisTrails pipeline workflow.*

Although ParaView is a general-purpose analysis and visualization framework that encompasses users from many scientific fields, UV-CDAT is focused primarily on climate data. Therefore, a simpler, domain-specific interface was created that exposed the features of ParaView that are most useful to the climate community. This includes such visualizations as contour plots and colormap plots.

In addition, several new climate-specific readers and filters have been implemented in ParaView for the UV-CDAT framework. These include an Unstructured Parallel Ocean Program (POP) reader, a Meridional Overturning Circulation (MOC) and Meridional Heat Transport (MHT) reader, and a Project Sphere filter. The Unstructured POP reader loads POP Ocean data as an unstructured grid. Better analysis and visualizations can be obtained from this more accurate representation of the data, as shown in Figures 28 and 29. The Project Sphere filter takes data defined on a spherical grid, like climate data, and projects it onto a flat plane. Support for computing the depth velocity from the horizontal velocity field was added to the vtkUnstructuredPOPReader. The POP code does not explicitly solve the vertical velocity; instead it computes from the divergence of the horizontal velocity field and the use of the continuity equation of fluid mechanics. The reader now computes this important derived field. A projection filter to go from a sphere to the Plate Carrée lat-lon projection was added. Cells that straddle the anti-meridian (180 degrees longitude) are split and tensors are transformed to the projected coordinate system.

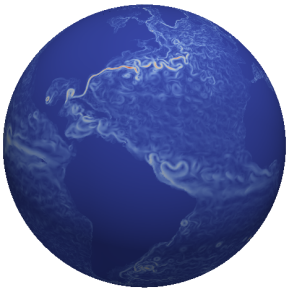


***Figure 27.*** *Visualization of velocity on a spherical representation of the Earth using vtkUnstructuredPOPReader.*

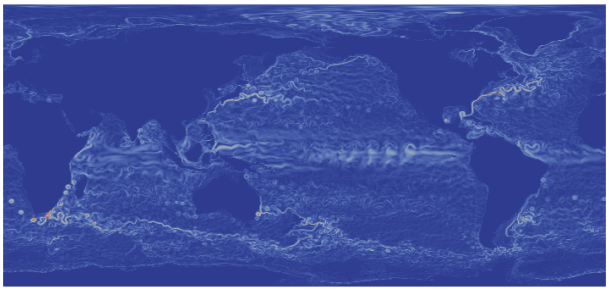


***Figure 28.*** *Visualization of velocity on a flat map (using Plate Carrée Projection).*

### *4.6.4 Meridional Overturning Circulation and Meridional Heat Transport*

The ocean transports a significant fraction of energy from the equator to pole in the climate system. Key measures of this transport are the MHT and MOC, often described by a meridional orverturning streamfunction (MOS). These are measures of how much heat and water are circulated poleward (equatorward) in both a global sense and in a specific basin (esp. the Atlantic). Because ocean model grids are not aligned with meridians and latitudes and transport quantities cannot be interpolated without introducing significant error, the computation of these two quantities requires some specialized computations that are difficult to perform efficiently.

Serial programs were first used to compute these two quantities. As the size of the generated data became ever larger, the time required to compute them became prohibitive, and they were dropped as part of the standard diagnostics. To remedy this, two parallel filters were created in ParaView and integrated into UV-CDAT, which calculate the MOC and MHT in parallel. The performance gains from parallelizing these computations were substantial, with run times being reduced from a couple of hours to a few minutes.

### *4.6.5 R*

The statistics package R is now integrated into UV-CDAT, currently utilized through VisIt and the VTK-R Bridge. This work has a two-fold benefit. One, analysis of climate data in R has sped up through parallelization, using VisIt's investments in parallel processing. Two, statistics and visualization routines can be used in combination, enabling analyses that were not possible before. This integration was accomplished in a way that allows for many R scripts to be supported. However, we also developed new R-based statistical approaches for analyzing climate data, and the team has deployed these approaches.

The current versions of VTK (>5.8) provide support for executing R code on VTK arrays. The R Interface layer of VTK handles the conversion of data from VTK to R data structures, and R to VTK. Building this data analysis capability on top of the mature and proven VisIt infrastructure allows for the parallel analysis and visualization that are critical to the climate community. We provide an implementation of a Generalized Extreme Value (GEV)

analysis with VisIt for the climate science community, which is described in detail within the VisIt section below. A chart of the scalability out to 256 processors where spatial and temporal parallelization was used is shown in Figure 30.

**Scaling study**

| num CPUs | Annual Analysis | Monthly Analysis |
|---|---|---|
| 1 | 600.53 | 7,904.97 |
| 2 | 320.95 | 5,954.16 |
| 4 | 181.40 | 4,948.32 |
| 8 | 112.20 | 2,589.61 |
| 16 | 67.49 | 2,121.38 |
| 32 | 34.36 | 1,317.06 |
| 64 | 19.51 | 673.61 |
| 128 | 10.33 | 558.18 |
| 256 | 5.99 | 576.66 |
| 512 | 5.10 | 317.34 |
| 1024 | 4.59 | 166.41 |

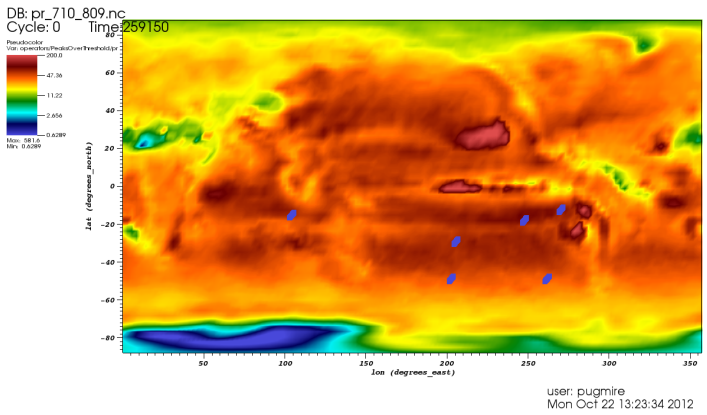



***Figure 29.*** *Scaling study and R execution. Left: Scalability of Extreme Value Analysis code. Right: Peaks Over Threshold of 100-year, daily precipitation CCSM3.0 data.*

Because R is built directly using UV-CDAT's build infrastructure, it can also provide functionality as a stand-alone executable that is integrated through VisTrails.

### *4.6.6 Visualization Control System*

The Visualization Control System (VCS) [Doutraiux, 2007] is the de facto standard 1D and 2D graphics package for UV-CDAT. It is especially designed for climate change research. Conforming to the netCDF CF convention, VCS allows users to have complete control over graphics. That is, by specification of the desired netCDF CF data, the graphics method, and the display template, the user can control the appearance of the data display, associated text, and animation. Because it is so flexible, wide ranges of graphical displays are predefined and users can create and share new ones. For the community of MIPs (including CMIP), users have predefined graphical output specific to their diagnostic climate modeling needs.

In the VCS model, the data display is defined by a trio of named object sets, designated the "primary objects" (or "primary elements"). These include the data, which define what is to be displayed and is ingested into the system via other UV-CDAT software components; the UV-CDAT plot type, which specifies the 1D or 2D display technique; and the picture template, which determines the appearance of each segment of the display. Tables for manipulating these primary objects are stored in VCS for later recall and possible use. In addition, detailed specifications of the primary objects' attributes are provided by eight "secondary objects" (or secondary elements"):

1. colormap: specification of combinations of 256 available colors.
2. fill area: style, style index, and color index.
3. format: specifications for converting numbers to display strings.
4. line: line type, width and color index.
5. list: a sequence of pairs of numerical and character values.
6. marker: marker type, size, and color index.
7. text: text font type, character spacing, expansion and color index.
8. text orientation: character height, angle, path, and horizontal/vertical alignment.

By combining primary and secondary objects in various ways (either at the command line or in a Python program), the VCS user can comprehensively diagnose and inter-compare climate model simulations. VCS provides numerous capabilities, including the ability to:

- Create and modify existing template attributes and graphics methods.
- Save a display as a GIF, Postscript, PNG, Encapsulated Postscript file, etc.
- Create and modify color maps.
- Zoom into a specified portion of a display.
- Change the orientation (portrait vs. landscape) or size (partial vs. full-screen) of a display.
- Animate a data variable.
- Display different map projections.

A VCS plot type simply defines how data is to be displayed on the screen. Currently, there are 13 different graphic methods with more on the way. Each graphic method has its own unique set of attributes (or members) and functions. They also have a set of core attributes that are common in all VCS plot types. The descriptions of the current set of plot types are as follows:

1. *Boxfill*—Draws color grid cells to represent the data on the graphics display.
2. *Isofill*—Fills the area between selected isolevels (levels of constant value) of a two-dimensional array with a user-specified color.
3. *Isoline*—Draws lines of constant value at specified levels in order to graphically represent a two-dimensional array. It also labels the values of these isolines on the graphics display.
4. *Meshfill*—Draws a two-dimensional data array by surrounding each data value by a colored grid mesh.
5. *Outfill*—Fills a set of integer values in any data array. Its primary purpose is to display continents by filling their area as defined by a surface type array that indicates land, ocean, and sea-ice points.
6. *Outline*—Outlines a set of integer values in any data array. Its primary purpose is to display continental outlines as defined by a surface type array that indicates land, ocean, and sea-ice points.
7. *Continents*—Draws a predefined, generic set of continental outlines in a longitude by latitude space. To draw continental outlines, no external data set is required.
8. *Scatt*er—*D*isplays a scatter plot of two 4-dimensional data arrays, e.g. A(x,y,z,t) and B(x,y,z,t).
9. *Taylordiagram*—Uses standard deviations, centered root mean square differences, and correlations to plot statistical variable differences in the comparison of model results to a reference model or to observations.
10. *Vector*—Displays a vector plot of a 2D vector field. Vectors are located at the coordinate locations and point in the direction of the data vector field. Vector magnitudes are the product of data vector field lengths and a scaling factor.
11. *Xvsy*—Displays a line plot from two 1D data arrays, that is X(t) and Y(t), where ‘t’ represents the 1D coordinate values.
12. *Xyvsy*—Displays a line plot from a 1D data array, i.e. a plot of X(y) where ‘y’ represents the 1D coordinate values.
13. *Yxvsx*—Displays a line plot from a 1D data array, i.e. a plot of Y(x) where ‘x’ represents the 1D coordinate values.

A picture template determines the location of each picture segment, the space to be allocated to it, and related properties relevant to its display. The picture template attributes describe where and how segments of a picture

will be displayed. The segments are graphical representations of textual identification of the data formatted values of single-valued dimensions and mean, maximum, and minimum data values axes, tick marks, labels, boxes, lines, and a legend that is plot-type specific to the data. Picture templates describe where to display all segments including the data.

Figure 31 below shows the many VCS plot types available in UV-CDAT.

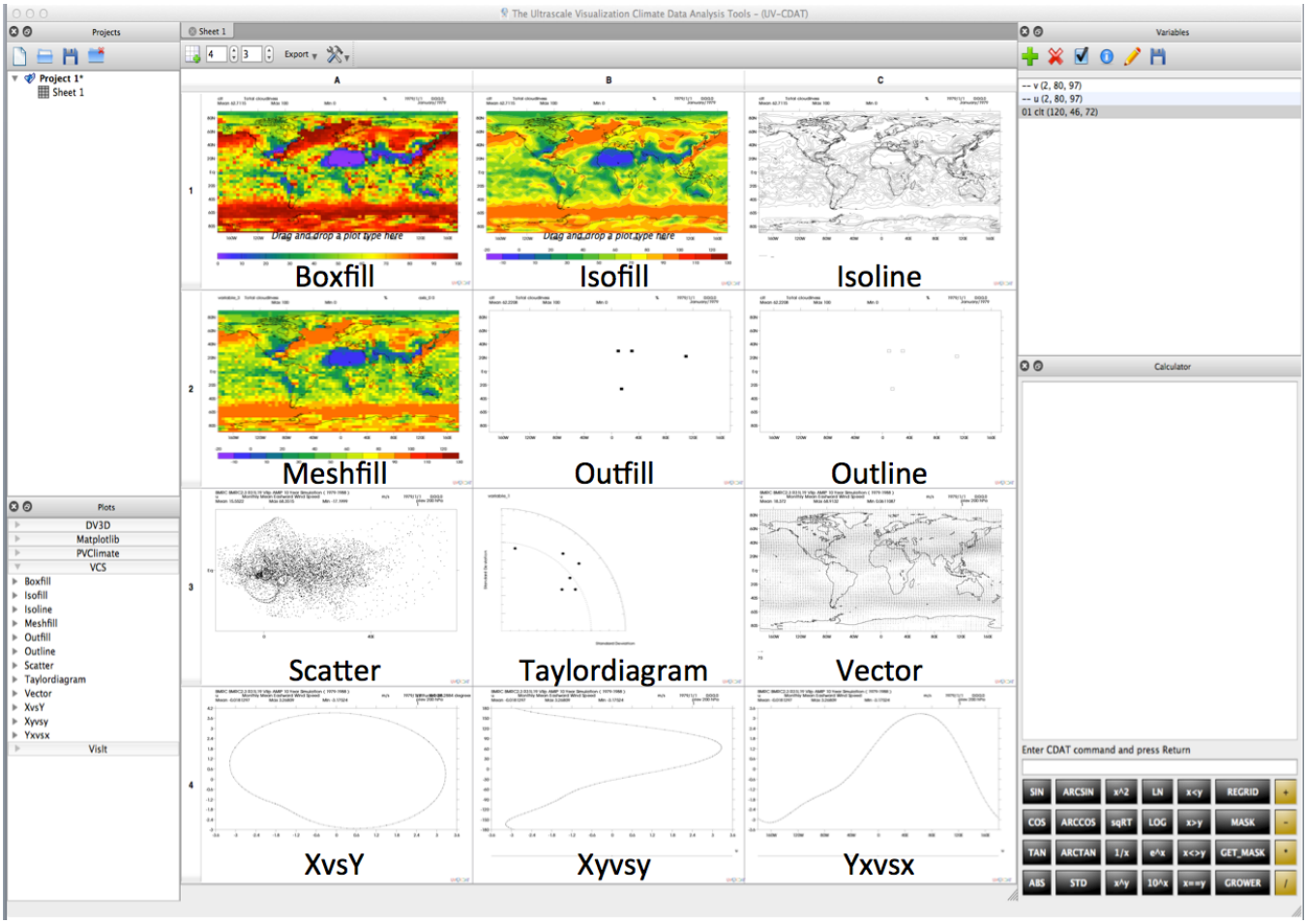



***Figure 30.*** *The image shows 12 of the 13 VCS plot types. From left-to-right, top-to-bottom, the visualization spreadsheet shows the following VCS plots: boxfill, isofill, isoline, meshfill, outfill, outline, scatter, taylordiagram, vector, xvsy, xyvsy, and yxvsx. A picture template defines each VCS image in visualization spreadsheet, plot type (e.g., boxfill, isofill, etc.), and data.*

### *4.6.7 VisIt*

The integration of VisIt brought not only the ability to utilize its rendering and analysis capabilities, but also several climate science operations developed by the Visual Data Explorer team. The addition of R for statistical operations, the modification of VTK to include VTK-R Bridge support, and the integration of VisIt into VisTrails are several key operations that were required for successful interoperability.

VisIt's license was upgraded to incorporate the inclusion of GPL licenses. This is due to the inclusion of GnuR statistics required to operate several climate specific algorithms. In addition to including GnuR support through the VTK-R Bridge, VisIt also added PyQt wrapping to its user interface, which was required to allow visual integration with other components in UV-CDAT such as ParaView, DV3D, and VCS within the VisTrails user interface (see Figure 4). VisIt's build infrastructure was also modified to work with external dependencies, specifically reusing libraries provided by other components. The most significant of these is VTK, which VisIt

was modified to share with ParaView. This also led to a significant size reduction of the VisIt binary, reducing its overall contribution from 100s of MB to approximately 30MB.

VisIt was added to UV-CDAT's internal build environment as well as the VisTrails user interface. The most significant addition is the integration of the VisIt client and the renderer into the UV-CDAT user interface. This addition adds the Python and GUI control layer, which effectively enables UV-CDAT users to manipulate VisIt just as if it was a standalone. The VisIt–VisTrails user interface module embeds several commonly used rendering operations.

VisIt was also extended to parallelize its processing temporally, as opposed to parallelizing spatially. Spatial parallelization is necessary for simulations in fusion, combustion, and other application areas, when there are billions of data points per time slice. Climate data is different, however, since a time slice of climate data is normally small enough to be processed serially. However, climate data normally has many time slices, and so temporal parallelization can provide a significant increase in speed. We demonstrated excellent scalability, adding more and more compute power, which led to proportional drops in processing time.

The Extreme-Value-Analysis and Peaks-Over-Threshold operations are advanced climate analysis routines whose operators in VisIt have been upgraded to include additional functionality and improved ease of use. In particular, users can now analyze ensembles, in which each ensemble member is treated as a separate data sample. This allows a user to decrease statistical uncertainty by combining runs in a statistically rigorous fashion to characterize the behavior of extremes in a climate model. In addition, results from analyses are now output as netCDF files. Finally, we have improved the GUI to make the operators easier to use. We are currently finalizing and testing the code. While the operators can handle basic analyses of point-by-point variation in return values and linear trends in return values over time, more sophisticated statistical analysis is possible through the new VisIt functionality for exercising general R and Python code.

### *4.6.8 ViSUS*

The ViSUS client within UV-CDAT enables simple and efficient access to massive data sets held at remote locations. Using a data reordering based on hierarchical space filling curves, ViSUS translates the user's local inputs in choosing the data set and variables of interest, viewpoint, visualization technique, etc. into small, remote data queries designed to require minimal space and compute resources to fulfill. The resulting data is then shipped to the client and processed locally. This approach has been shown to provide an efficient and scalable data access that supports networks ranging from high-end backbone to public Wifi connections and from top-of-the-line workstations to mobile devices such as cell phones or tablet computers.

Hierarchical indexing was originally introduced for out-of-core data accesses to avoid unnecessary file I/O. Out-of-core computing addresses the issues of algorithm redesign and data layout restructuring that are necessary to enable data access patterns having minimal out-of-core processing performance degradation. Research in this area is also valuable in parallel and distributed computing, where one has to deal with the similar issue of balancing processing time with the time required for data access and movement among elements of a distributed or parallel application. The solution to the out-of-core processing problem is typically divided into two parts: (1) algorithm analysis, to understand data access patterns and, when possible, redesign to maximize data locality, and (2) storage of data in secondary memory using a layout consistent with the access patterns of the algorithm, amortizing the cost of individual I/O operations over several memory access operations.

In the case of hierarchical visualization algorithms for volumetric data, the 3D input hierarchy is traversed from a coarse grid to the fine grid levels to build derived geometric models having adaptive levels of detail. The shape of the output models is then modified dynamically with incremental updates of their level of detail. The parameters that govern this continuous modification of the output geometry are dependent on runtime user interaction, making it impossible to determine, a priori, what levels of detail will be constructed. For example, parameters can be external, such as the viewpoint of the current display window or internal, such as the isovalue of a contour or the position of a slice plane.

The general structure of the access pattern can be summarized into two main requirements: (1) the input hierarchy is traversed from coarse to fine and level by level so that data in the same level of resolution is accessed at the same time, and (2) within each level of resolution, the regions that are in close geometric proximity are stored as much as possible in close memory locations and also traversed at the same time.

The Lebesgue space filling curve, also called Z-order space filling curve for its shape in the 2D case, is depicted in Figure 32. The Z-order space filling curve can be defined inductively by a base Z shape of size 1 (a), that is by the vertices of a square of side 1 that are connected along a "Z" pattern. Such vertices can then be replaced each by a Z shape of size 1/2 as in (b). The vertices obtained in this way are then replaced by Z shapes of size 1 as in (c), and so on. In general, the ith level of resolution is defined as the curve obtained by replacing the vertices of the (i − 1)th level of resolution with Z shapes of size 1/2i . The 3D version of this space filling curve has the same hierarchical structure, with the only difference being that the basic Z shape is replaced by a connected pair of Z shapes lying on the opposite faces of a cube as shown in Figure (f). Figure 32 (f-j) show five successive refinements of the 3D Lebesgue space filling curve. The d-dimensional version of the space filling curve has also the same hierarchical structure, where the basic shape (the Z of the 2D case) is defined as a connected pair of (d − 1)-dimensional basic shapes lying on the opposite faces of a d-dimensional cube.

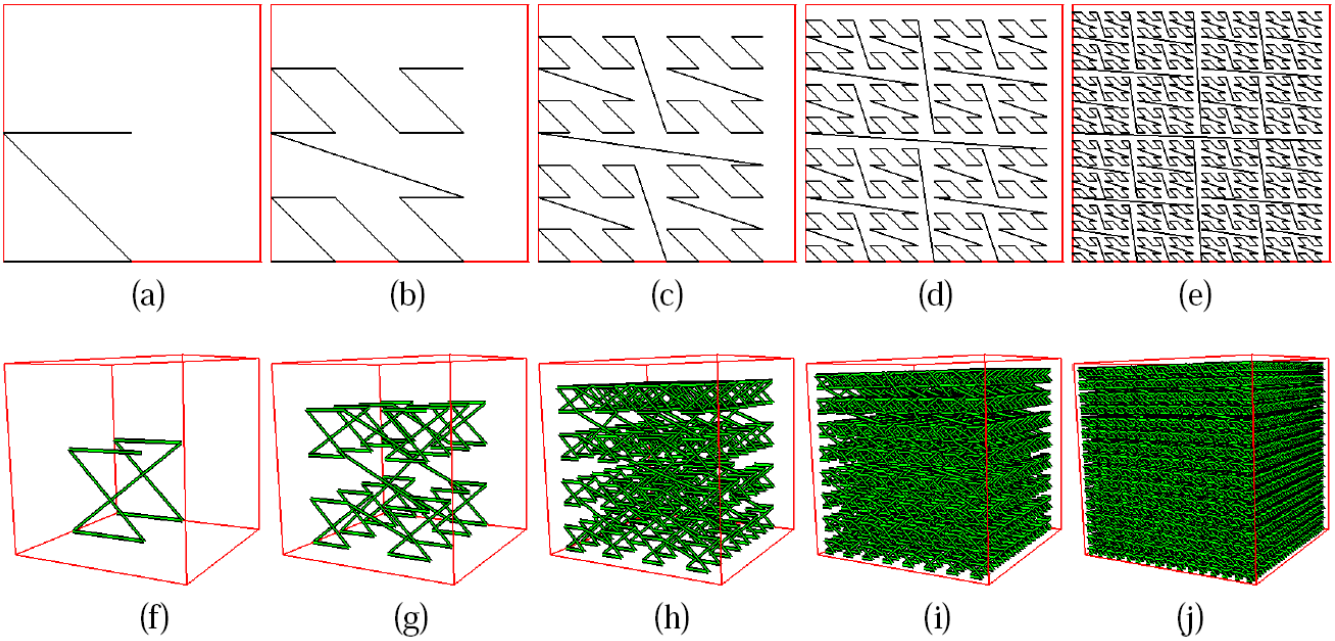



***Figure 31 (a-j).*** *The first five levels of resolution of the 2D Lebesgue's space-filling curve. (f-j) The first five levels of resolution of the 3D Lebesgue's space-filling curve.*

ViSUS uses a static indexing scheme that induces a data layout satisfying both requirements (1) and (2) above for the hierarchical traversal of n-dimensional regular grids. The scheme has three key features that make it particularly attractive. First, the order of the data is independent of the out-of-core block structure, so that its use in different settings (e.g. local disk access or transmission over a network) does not require any large data reorganization. Second, conversion from the Z-order indexing—see Figure 32—used in classical database approaches to this indexing scheme can be implemented with a simple sequence of bit-string manipulations, making it appealing for a possible hardware implementation. Third, since there is no data replication, it avoids the performance penalties associated with dynamic updates as well as increased storage requirements typically associated with most hierarchical and out-of-core schemes. Beyond the theoretical interest in developing hierarchical indexing schemes for n-dimensional space filling curves, this approach targets practical applications in out-of-core visualization algorithms.

The 1D order can be structured in a binary tree by considering elements of level i, those that have the last i bits all equal to 0. This yields a hierarchy where each level of resolution has twice as many points as the previous level. From a geometric point of view, this means that the density of the points in the d-dimensional grid is doubled alternating along each coordinate axis. Figure 33 shows the binary hierarchy in the 2D case where the resolution of the space-filling curve is doubled alternately along the x- and y-axis. The coarsest level (a) is a single point, the second level (b) has two points, the third level (c) has four points (forming the Z shape), and so on.

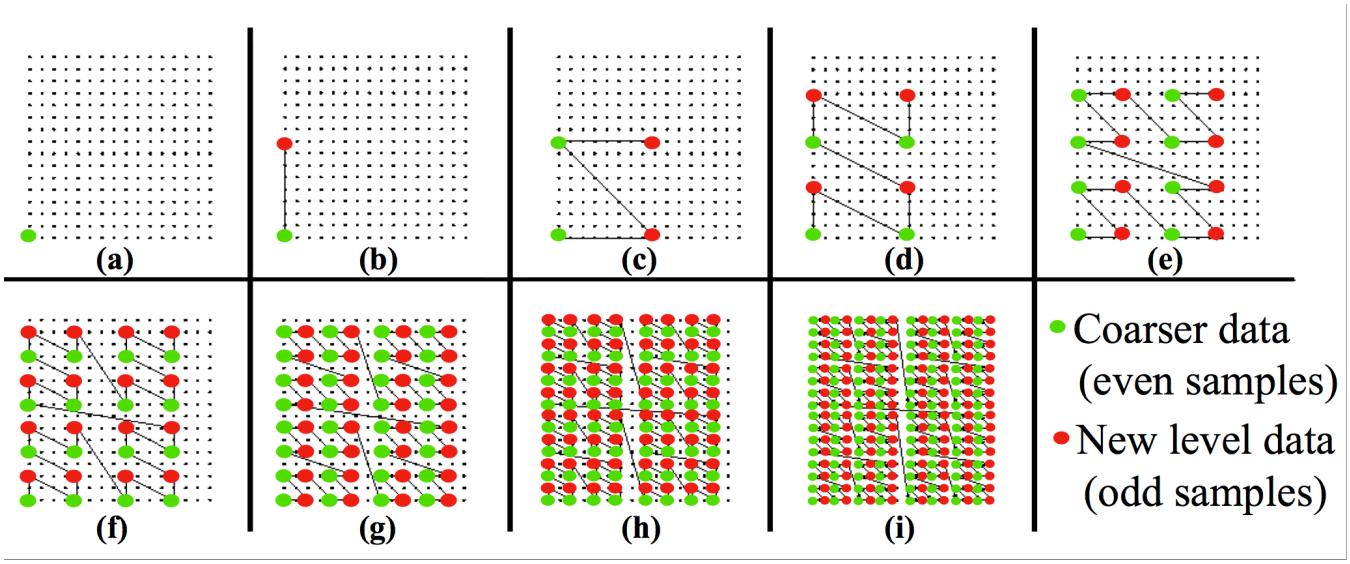



***Figure 32 (a-i).*** *The nine levels of resolution of the binary tree hierarchy defined by the 2D space filling curve applied on 16 × 16 rectilinear grid. The coarsest level of resolution (a) is a single point. The number of points that belong to the curve at any level of resolution (b) to (i) is double the number of points of the previous level.*

Given today's massive data sets, even simple manipulations can be overly expensive if they must be applied to each variable separately. For data of this size, it would be ideal to process the data as it is requested and pushed to a user and most ideally only operate on the portion of the data needed for display. The multi-resolution data layout outlined above enables the efficient access of a subset of a large data set. Moreover, this data will be accessed in a progressive hierarchy. Operations such as binning, clustering, or rescaling are trivial to implement on this hierarchy given some known statistics on the data, such as the function value range, etc. These operators can be applied to the data stream as is, while the data is moving to the user, progressively refining the operation as more data arrives. Much more complex operations can be reformulated to work well using the hierarchy. For instance, using the layout for 2D image data produces a hierarchy that is identical to a sub-sampled image pyramid on the data. Moreover, as data are requested, the progressivity of the transfer will traverse this pyramid in a coarse-to-fine manner. Techniques such as gradient-domain image editing can be reformulated to use this progressive stream and produce visually acceptable solutions. These adaptive, progressive solutions allow the user to explore a full resolution solution as if it was fully available, without its expensive computation. Therefore, the hierarchical Z-order is the layout of choice for efficient data management for visualizing and analyzing large scale scientific simulation data.

This technology is a crucial next step in climate visualization as data files are increasingly stored at remote locations and are far too big to be easily transferred. Furthermore, beyond simple data accesses ViSUS has been extended to support a broad range of server-side processing such as on-the-fly averaging, comparing, or otherwise transform multiple data sets. This is especially useful in climate science as one of the most common tasks in any climate analysis is to aggregate multiple data sets, for example, to compute an ensemble's mean or standard deviation. Typically, this is an expensive operation requiring both significant compute as well as storage resources and often means data must be kept locally. Instead, UV-CDAT through the ViSUS client now allows users to

specify multiple data sets as well as a host of potential operations, e.g., mean, standard deviation, etc., and have the remote server simultaneously query and process all data sets, sending only the results to be visualized. This provides a new level of flexibility to explore both single data sets as well as large ensembles held remotely, with the potential to significantly accelerate the current analysis workflow.

### *4.6.9 Visualization Toolkit (Kitware)*

The Visualization Toolkit (VTK) is an open-source, freely available software library for 3D computer graphics, image processing, and visualization. VTK consists of a C++ library and several interpreted layers including Tcl/Tk, Java, and Python. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods, and advanced modeling techniques such as implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. Additionally, VTK has a suite of 3D interaction widgets and supports parallel processing and integration with various databases. VTK is cross-platform and runs on Linux, Windows, and Mac platforms. Due to the organization of algorithms into pipelines, VTK provides an exceptionally flexible methodology for producing, modifying, and re-configuring visualization; maximizing code reuse and customization, while maintaining scalability, efficiency and flexibility. VTK is used worldwide for academic, research, and commercial applications. UV-CDAT utilizes VTK extensively. ParaView, VisIt, and DV3D all use VTK as their underlying visualization library. VTK's ability to support numerous data formats, grid types (see Figure 34) and visualization techniques makes it an ideal candidate to support visualization needs of the climate science community.

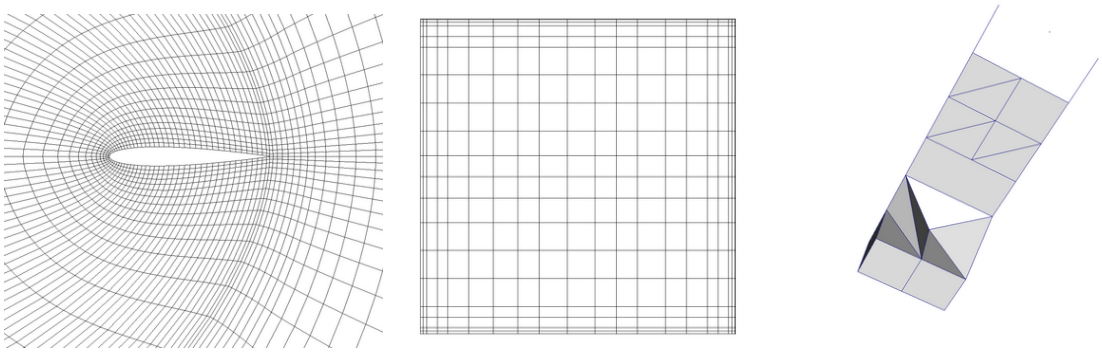


**Figure 33.** *VTK supports various kinds of grids such as rectilinear, curvilinear, and unstructured.*

Also, in collaboration with Sandia National Laboratories, Kitware developed the Titan informatics toolkit, which significantly expanded the informatics capabilities of VTK. As part of this effort, the ability to display and project geospatial information has been developed in the form of a Geovis API. 2D and Charts API (built on top of 2D API) of VTK provide the ability to render interactive views such as graphs and hierarchical trees, and plots such as scatter plots, pie charts, and flood plots, as shown in Figure 35. We have begun to leverage some of these capabilities in UV-CDAT.

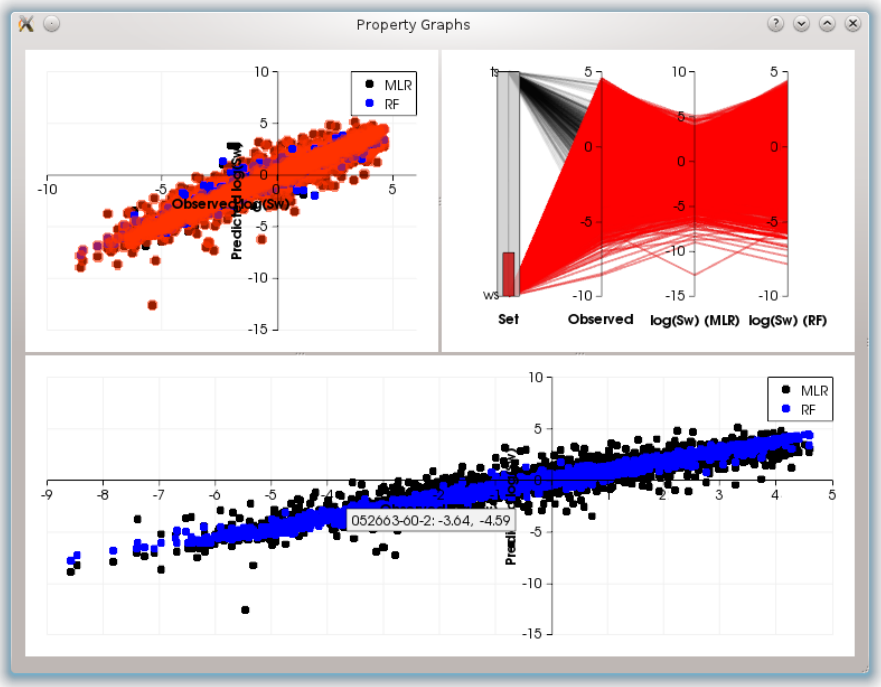



***Figure 34.*** *VTK supports multiple types of 2D and 3D plots with linking and brushing capabilities.*

## 4.7 Diagnostic Packages

Researchers are uneasy when it comes to sharing their diagnostics. They spend a lot of time putting them together to prove a point, to further the science, or for publication. However, in order to further prove or disprove climate phenomena, others must use these diagnostics. Eventually, therefore, bits and pieces of diagnostic codes must be (and are) exchanged with others in collaboration. Moreover, the process of "adopting" a new diagnostic is an arduous process on the part of both the diagnostic provider and the user. A harmful consequence of releasing diagnostics piecemeal is that multiple or similar (but not identical) versions of diagnostics are used, which produce slightly different results.

Currently, diagnostics are being continuously redeveloped to meet a specific I/O format, programming language, or simply because the user doesn't trust external code. UV-CDAT is trying to close that gap in diagnostic trust and reuse by providing a common environment in which all researchers can develop, encourage, facilitate communication and sharing in diagnostic development.

For the DOE- ESM project, several packages will run automated and interactive diagnostics on the model component simulation output for atmospheric, land, ocean, and sea-ice. UV-CDAT scripts have been created to generate data files required to interactively produce multiple diagnostics and plots that are substantially equivalent to the batch output of the NetCDF Operators (NCO) AMWG diagnostics package. Unlike the NCO AMWG package with static plots, the UV-CDAT AMWG version allows for dynamic user interaction and data queries of the ESGF distributed archive. Interactively, the UV-CDAT functions allow such changes as contour intervals and colors, zoom to regional locations, and displaying multiple user selected plots on one or more canvas displays.

Over the longer term, the diagnostics will be expanded to not only cover the Community Atmosphere Model (CAM) but also include the ocean, land, and polar climate model working groups in FY14. UV-CDAT's climatological and departures capabilities are used to generate the means of the simulations, and its plot capabilities are used to produce plots and tables of mean climate in the NetCDF format and CF convention.

### 4.7.1 Atmospheric Model Working Group (AMWG)

CAM5, the atmospheric component of the Community Earth System Model (CESM), produces thousands of output fields, which can vary in time, latitude, longitude, and height. Gleaning climate predictions or evaluating model skill from this output requires analysis of a myriad of metrics derived (often via complex calculations) from these fields. To do this, modelers have come to rely on a monolithic shell script combined with NCL routines known collectively as the "AMWG diagnostics package." This script is hard to follow, generates hundreds of plots that never get examined, and lacks the flexibility to create new plots to test new ideas and data. We are developing an alternative package, which avoids these issues. First we compute climatological means and departures, which are stored to a distributed archive (i.e., ESGF) for interactive manipulation in a format optimal for analysis with UV-CDAT. Scripts have been added to UV-CDAT to automatically plot quantities typically used by climate modelers as well as to enable more sophisticated data exploration.

### 4.7.2 Land Model Working Group (LMWG)

The land model diagnostics share many of the same requirements as those of the AMWG and hence build upon the same core underlying diagnostics framework within UV-CDAT. The LMWG is designed to support detailed analysis of climatologies and departure means, similar to the AMWG. The LMWG must also support comparison between two global simulations in the areas of carbon fluxes, carbon states, hydrology, and drivers. Our work to date has focused on developing a base set of standard diagnostic routines, which generate summary data sets that can then be further analyzed/visualized by the end user or automatically processed as part of a larger workflow into a variety of value-added data products.

### 4.7.3 Ocean Model Working Group (OMWG)

While many diagnostics are based on surface fields or standard slices, some ocean diagnostics are more difficult to compute, especially as the ocean model utilizes grids that are not aligned with standard coordinates. In particular, transport diagnostics are computed across straits or observational sections and the meridional diagnostics (overturning streamfunction and heat transport) require more compute-intensive analysis. The MOC was identified as a POP) diagnostic that was dropped by the POP users at LANL when they moved to a high-resolution, 1/10 degree grid because the diagnostic was consuming too much compute time at that resolution. The diagnostic was then produced only when absolutely necessary, by post-processing of prognostic fields using the same poorly performing code with the added cost of I/O. The UV-CDAT project targeted this diagnostic of the OMWG diagnostics package standard for parallelization making a substantial impact on the time to solution. This included support for LANL-specific POP implementation of Partial Bottom Cells.

A related diagnostic, MHT, was also speeded up and is computed simultaneously at very little cost. Speedups of serial versions are greater than 60x, and parallel versions are substantially faster depending on I/O read performance, taking less than a second to calculate the MOC when the requisite prognostics are in memory. The LANL POP team was pleased with the post-processing performance gains, and is currently working to reestablish this new parallel MOC and MHT as part of the LANL POP runtime diagnostic standard. During the development of this new diagnostic routine, the implementation assumed an unstructured grid equivalent to POP's structured grid. This change anticipated the move toward the new Model for Prediction Across Scales ocean model that will soon replace POP as the ocean model component.

### 4.7.4 Polar Climate Working Group (PCWG)

The PCWG set of standard diagnostics are focused on analysis of sea ice, including sea ice extent and volume, ice velocities, and many other fields. Because the sea ice utilizes the same grid as the ocean, many of the changes made during the UV-CDAT project (e.g., POP readers, ocean diagnostics) also benefit the sea ice model diagnostics. This has been useful in the analysis of high-resolution coupled simulations being performed by DOE researchers.

In addition, a new MPAS-based sea ice model is being developed to work with the MPAS-Ocean model, requiring unstructured grid diagnostics. The PCWG diagnostics make use of polar projections in most visualizations and will benefit from new projection and regridding capabilities. Finally, new diagnostics are being developed to take advantage of both new diagnostic fields (e.g., ice age, ridging) and new observations. These will continue to be incorporated into new standard diagnostics packages and new MPAS-based diagnostics under future funded efforts.

## 4.8 Metrics

The World Climate Research Programme (WCRP) has established a panel to foster the development of routine performance metrics for climate models. This panel, jointly established by the Working Group on Numerical Experimentation (WGNE) and the Working Group on Coupled Models (WGCM), has developed a package to quantify model errors (based on comparison with observations) and to assess the relative model performance of different models. The WGNE/WGCM metrics package was designed with UV-CDAT, which is ideally suited for this purpose, given its breadth of analysis capabilities. The package includes routines for computing standard metrics, carefully selected observation data sets, and a database of results from all models contributed to CMIP3 and CMIP5. The package is currently being evaluated by several modeling groups and will be widely distributed in September 2013.

In the more than 20 years of WCRP model intercomparison projects, modeling groups have contributed their data with little information on how well their model compares to other state-of-the-art models. With the WGNE/WGCM metrics package, modeling groups will be able to test new versions of their model (see Figure 36) against others *during* the model development process. As the modeling community begins to work towards a CMIP6, the WGNE/WGCM metrics panel intends to broaden this model-data interrogation to include a diverse range of community-based metrics for Earth System Models.

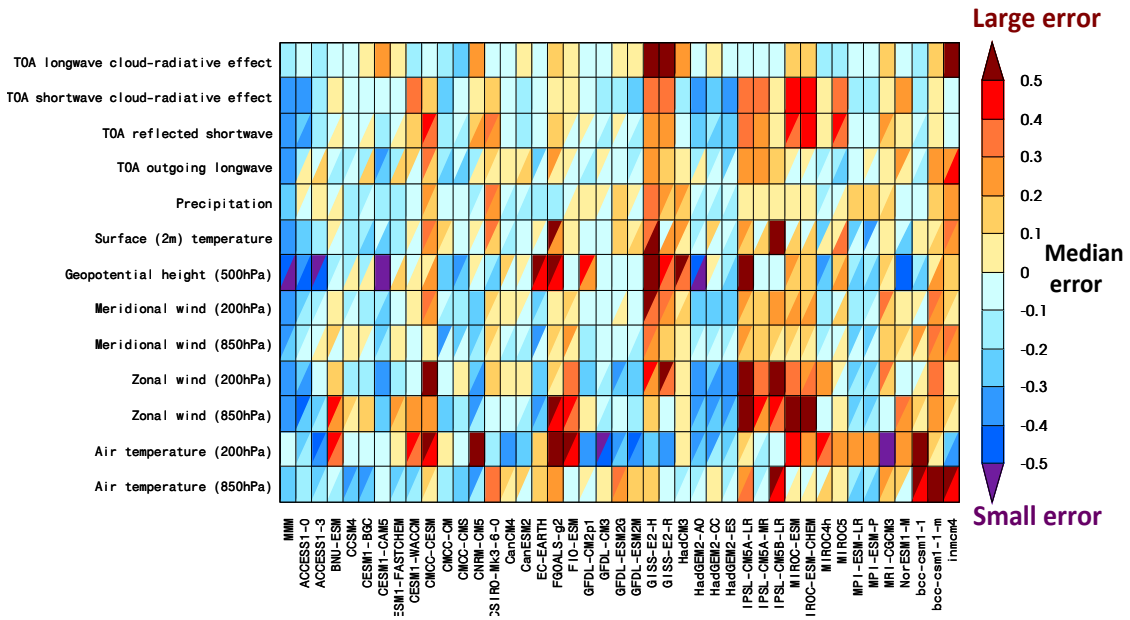
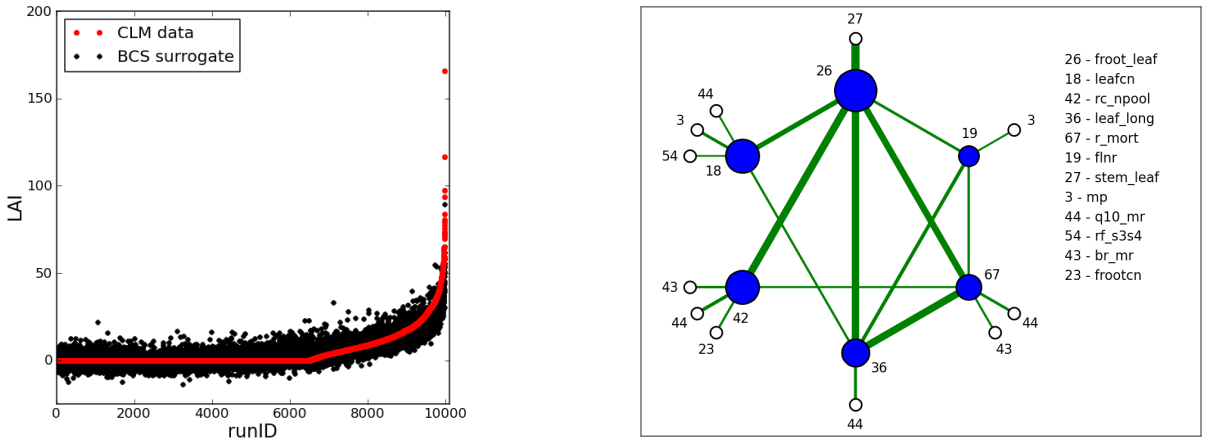



***Figure 35.*** *Portrait diagram depicting the root mean-square error (RMSE) of the annual cycle for the CMIP5 Multi-Model Mean (MMM) and individual contributing models (columns), for each of a selection of atmospheric metrics (rows). A normalized value of positive (negative) 0.5 indicates an RMSE 50% larger (smaller) than the MMM. For each entry, the upper left (lower right) triangles represent the model RMSE with respect to the following default (alternate) observational reference data sets: ERA-INT (NCEP1-RNL) for upper air fields; CERES-EBAF (ERBE) for radiation; and GPCP (CMAP) for precipitation. For additional discussion of the portrait plot, see Gleckler et al. (2008).*

## 4.9 Uncertainty Quantification (UQ)

We have used Python scripts to wrap select C++/Python tools from the Uncertainty Quantification Toolkit (UQTk) (http://www.sandia.gov/UQToolkit) and make them accessible to UV-CDAT. The UQTk is a software library that facilitates the propagation of uncertain inputs and parameters through a computational model by using Polynomial Chaos expansions for random variables and stochastic processes. The tools connected to UV-CDAT include the recently developed iterative Bayesian compressive sensing algorithm [Sargsyan, 2013], which was used in conjunction with polynomial chaos expansions to select the best basis sets and construct inexpensive surrogates for computationally expensive models. The surrogate models are used, without much computational overhead, in place of the expensive model, in global sensitivity studies for observables of interest. Figure 37 displays data generated by UQTk tools accessed from within the UV-CDAT framework.




a) datbcs.png b) senscirc.png

***Figure 36.*** *UQ output integrated into UV-CDAT with the help of Sandia National Laboratory. a) datbcs.png shows the distribution of surrogate model evaluations for Leaf Area Index (LAI), which is one of the observables generated by the CLM. b) "senscirc.png" shows the most important CLM model parameters, with blue circles, and combination of parameters, with green lines, contributing to the variance in LAI. The circle sizes and line thicknesses are proportional to the importance of these parameters to LAI variance.*

## 4.10 Graphical User Interface

The UV-CDAT Graphical User Interface (GUI), shown in Figure 38, is designed to be easy to use but still support a broad range of data exploration tasks. The building blocks for creating visualizations are variables and plots. Variables represent the data to analyze and can link to local files, remote files, or any of the other available data sources. Plots are methods for visualizing data. UV-CDAT contains a large selection of plot types for visualizing climate data.

The basic usage is: First drag a variable to a spreadsheet cell, then drag a plot type to the same cell. The cell will then render the selected data with the selected plot type. Variables and plots can be dragged into a cell in any order. Using a queue system, variables are matched to plots based on the order they were dropped.

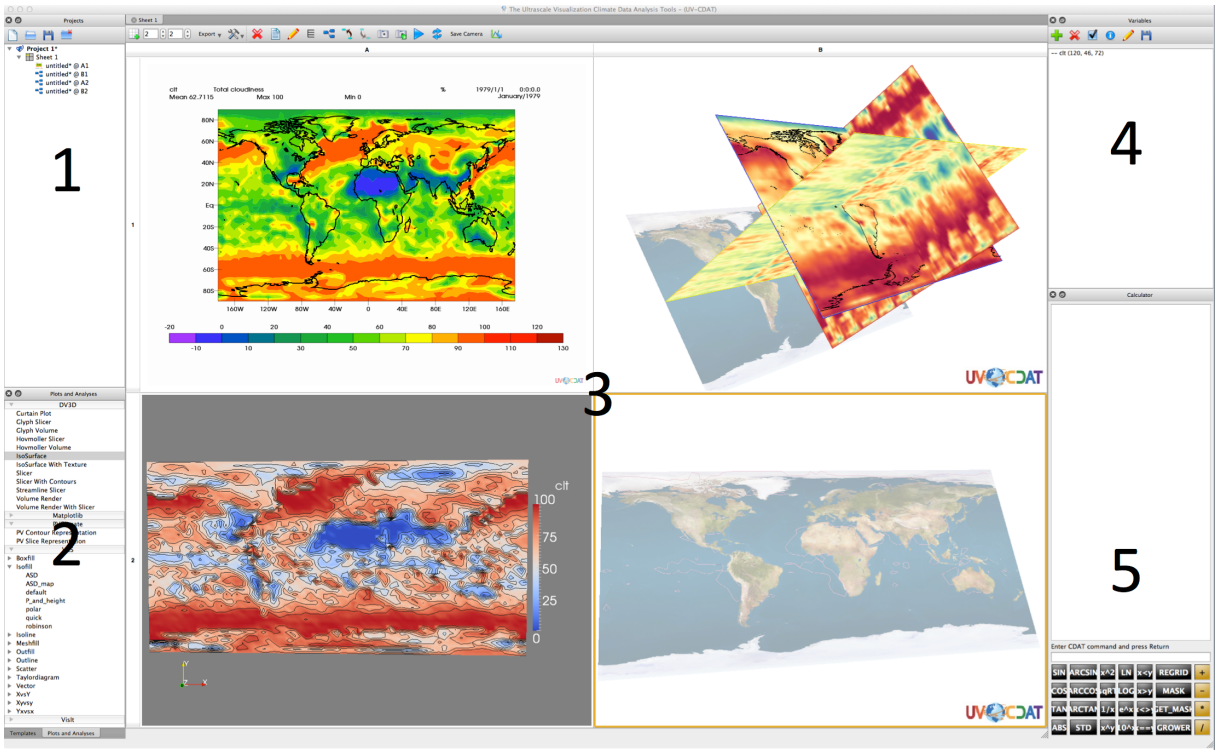



***Figure 37**. The Graphical User Interface is divided into many primary parts: 1) project; 2) plot types and templates; 3) visualization spreadsheet; 4) variables; 5) calculator; the main menu (not displayed); and VisTrails (also not displayed).*

### *4.10.1 Projects*

A project is a collection of variables, tagged visualizations, and spreadsheets. The project widget lists open projects along with their sheets and visualizations in a collapsible/expandable directory-like view. By default, it is docked in the upper left. Each project can have multiple spreadsheets, each with multiple visualizations. A visualization in a cell can be tagged and reused in other cells or spreadsheets. All sheets, tagged visualizations, and variables are saved with the project, along with all of the provenance information automatically captured by VisTrails. The project widget also contains a toolbar, with buttons to create, load, save, and close projects. These actions are also available under the File dropdown of the main menu.

### *4.10.2 Plots Types and Templates*

A plot is a method for visualizing data. The plot widget contains an expandable/collapsible list of available plots and is docked in the bottom left by default. Plots are grouped into packages, which are based on the technology used to render them. For example, the Matplotlib package contains plots that use the Matplotlib library to render the visualization. Clicking on the plot package expands its section revealing all of the plots for that package. Some packages, such as VCS, further categorize their plots based on the specific underlying algorithm used. These plots can then be dragged into a cell on the spreadsheet and combined with other variables and plots to make meaningful visualizations.

Each plot takes one or more variables as inputs and has various parameters that can be adjusted using the plot properties panel to change the final result. Many plots also allow users to interactively explore specific parts of the visualization using their computer mouse. Several features exist to make plot comparison easier. For example, users are able to sync certain parameters between similar plots, like camera angle, and some plots can be overlaid atop one another or in different layouts using templates.

Templates are used to determine the layout of plots within a single spreadsheet cell. The default behavior when multiple plots are combined in a single cell is to overlay multiple plots atop one another. Templates can be used to

display the plots side-by-side, among other configurations, in a single cell. The templates widget is docked in a tab behind the plots widget in the bottom left by default. It contains a list of all available templates, each of which can be dragged into spreadsheet cells just like plots and variables.

After a plot has been combined with one or more variables, the plot properties panel can be displayed and used to change the final result. This is done by either selecting View → Visualization Properties from the main menu or by clicking the pencil icon in the spreadsheet toolbar.

### *4.10.3 Visualization Spreadsheet*

The spreadsheet (middle), which is based on the VisTrails spreadsheet, is a resizable grid where each cell can hold a visualization. Users drag variables and plots to cells in the spreadsheet. The spreadsheet enables visualizations to be compared and explored in groups. Using cell synching, it is possible to modify multiple visualizations simultaneously, such as camera angle or cross-section sliders. The spreadsheet also supports undo/redo using the change-based provenance provided by VisTrails. The top of the spreadsheet contains the column headers, which are labeled and ordered alphabetically from A–Z. Clicking on a column header selects all of the cells in that column. The left side of the spreadsheet contains all of the row headers, which are labeled and ordered numerically starting at 1. Clicking on a row header selects all of the cells in that row. Clicking in the top left corner—the intersection of the row and column headers—selects all cells in the spreadsheet.

Above the spreadsheet column headers lies a toolbar that contains several tools pertaining to the spreadsheet itself, as well as the visualizations it contains. Many of these toolbar buttons only appear or are only usable after a visualization has been created and selected in the spreadsheet. These toolbar buttons include:

- **New Sheet**: adds a new empty spreadsheet in a new tab to the project.
- **Rows**: changes the number of visible rows in the spreadsheet.
- **Columns**: changes the number of visible columns in the spreadsheet.
- **Export**: displays actions for exporting a visualization as a single image, or each cell individually.
- **Setting**: offers several adjustable settings, including icon theme and auto-exec.
- **Clear**: clears the selected cells, allowing new visualizations to be created in their place.
- **Script**: brings up a text widget that displays the Python script used to generate the visualization with options to save the script.
- **Edit**: brings up the Plot Properties widget, which can be used to change various parameters that affect the resulting visualization.
- **Queue**: brings up the queue manager that displays the plots and variables currently in the queue for a cell,
- **VisTrails**: brings up the VisTrails Builder window that displays the underlying pipeline and provenance, among other things,
- **Undo**: undoes last action in selected cell.
- **Redo**: redoes last action in selected cell,
- **Dimension**: Consists of three buttons and allows users to traverse forwards or backwards through dimensions of high-dimensional data sets. For example, it can be used to cycle through the time dimension for plots that can only display one time step at a time.
- **Print**: prints the current plot.
- **Save**: similar to export, saves plot as an image.
- **Colormap**: edits the color table for a visualization.

- **Animate**: brings up the animation widget.

### 4.10.4 Variables

Variables represent the data to explore. Variables can be loaded either from local files or from one of the supported remote data sources. These include ESGF and NASA's DODS Catalog via OpenDAP. Loaded variables are listed in the variable widget, which is docked in the upper right by default. The variable widget also contains a toolbar with various buttons for acting on the variables. These include:

- **Load**: brings up the Load Variable widget.
- **Delete**: prompts the user to delete selected variables.
- **Select all variables**: selects all variables in the list.
- **Information**: displays detailed information about a variable.
- **Edit**: brings up the edit variable widget.
- **Save**: saves the variable to a netCDF file.

The variable edit panel allows users to select subregions by either drawing a rectangle on a map around the region of interest or by using the slider widgets to set boundaries on each dimension individually. There is also a calculator window (lower right) for performing simple calculations on variables such as getting the difference between two data sets or performing a regrid operation. Additional climate related tools and operations are available from the Tools dropdown menu.

### 4.10.5 Calculator

The calculator widget consists of an output window at the top, followed by a command line input, and a grid of buttons below. It is docked in the bottom right by default. Users can type commands using variable names, drag variables into the command line, or use the buttons to perform operations on selected variables. Any newly generated variables from these operations automatically appear in the variables widget and can be used to make new plots. Additionally, various raw Python commands can be entered here.

### 4.10.6 VisTrails

Users can bring up the full VisTrails window for any plot, allowing them to view detailed provenance information related to all user actions and workflow executions that were performed. The workflow generated for each plot can be edited here as well, allowing advanced users to tweak plots in ways that might not be available from the UV-CDAT GUI. Many more actions are available from the VisTrails window. Visit www.vistrails.org for more information.

### 4.10.7 Main Menu

UV-CDAT's main menu is divided into six sections. See Appendix E for more details.

## 4.11 Scripting

UV-CDAT allows scripts to call workflows with different parameters from a scripting environment. To refine this, we added an export capability where each module can specify a translation of itself to a scripting command. Finally, we have developed scripting methods that allow users to build workflows using a scripting syntax. For example, constructing an object adds a module to a workflow, and using the result of a call on an object when constructing another object creates a connection between the two modules. This approach has been successfully used to translate VTK scripts to workflow. For UV-CDAT scripts, it is often necessary to slightly modify the original scripts to make them more easily translatable to workflows (see Figure 39). One problem with this

approach is that some calls are interpreted immediately in Python while others only modify a workflow that is executed later.

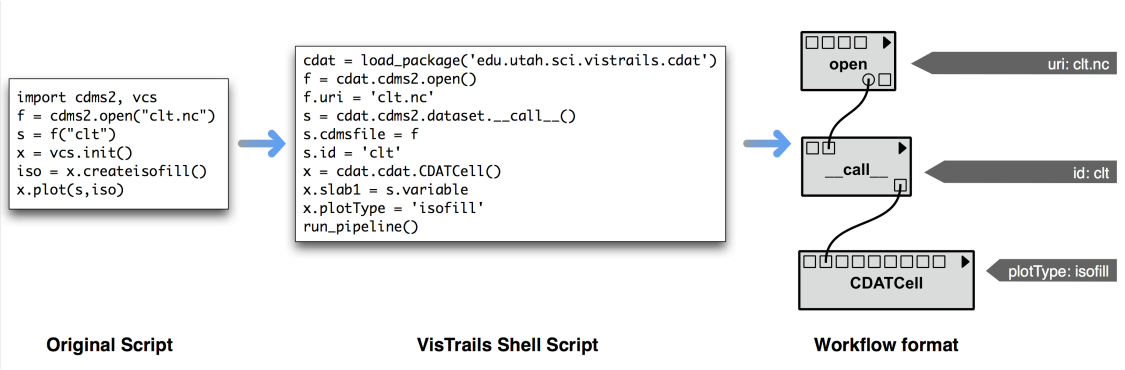



***Figure 38:*** *Simple UV-CDAT Python script opening a file and plotting a 2 dimensional contour plot*

We are currently working on a hybrid approach where certain calls are interpreted and translated to workflow constructions while others are maintained as Python source code in a PythonSource module (already available in UV-CDAT). A key consideration here is that modules are well correlated with existing lower-level calls for packages in UV-CDAT like CDMS and DV3D. This is important for packages that have scripting support from previous versions like CDMS because we wish to minimize translation. This approach will allow users to move projects between the UV-CDAT GUI and scripting environments while maintaining a single execution path that is provenance-enabled.

## 4.12 Workflow and Provenance Generation

### *4.12.1 Building UV-CDAT VisTrails Packages*

A key ingredient in integrating new and existing functionality into UV-CDAT workflows was creating a VisTrails package that encapsulates the methods and operations. A VisTrails package is similar to a Python package, and defines a set of modules (VisTrails classes) and I/O ports for each module for a particular interface. These modules are used by VisTrails to build workflows using UV-CDAT functionality.

The UV-CDAT VisTrails packages can be either manually written or automatically generated using Python's introspection and/or Extension Markup Language (XML) descriptions of existing functions (e.g., from the CDAT Python modules). For automatic generation, work has been done to:

- Enable all functions in UV-CDAT where introspection is not available to generate XML descriptions.
- Make sure they are correctly parsed and the UV-CDAT packages are functional in VisTrails.

UV-CDAT VisTrails packages can also be created from scratch (e.g., DV3D); such packages encode all functionality as part of the package code since they do not rely on an existing library. Currently, UV-CDAT supports VisTrails packages for CDAT, VCS, ParaView, DV3D, and Matplotlib.

### *4.12.2 Pipeline Helpers*

Pipeline helpers are important components in translating from the UV-CDAT GUI to underlying workflows. They provide the means both for translating changes in variables or plots to workflows as well as gluing the variable derivation to the plot construction. We have provided a base pipeline helper class that helps developers to persist changes in the GUI to the workflow. For example, this class provides methods for locating modules by type or the output port corresponding to a specific derived variable.

### 4.12.3 Provenance Capture

UV-CDAT captures both the prospective and retrospective provenance of analysis and visualization tasks from the VisTrails workflow and provenance infrastructure. Because VisTrails captures the evolution of workflows (the changes to the specification of a workflow), users can examine this prospective provenance to see how a particular analysis was constructed and refined. In addition, VisTrails automatically captures retrospective provenance as a workflow executes. Thus, for any analysis or visualization designed and executed in UV-CDAT, provenance information is captured automatically.

### 4.12.4 Provenance (PROV) Support

The provenance community has been working to standardize formats for generic provenance information. Building on the Open Provenance Model (OPM), the World Wide Web Consortium (W3C) has released the Provenance (PROV) standard for modeling and interchanging provenance information. VisTrails has integrated support for both OPM and PROV so UV-CDAT also is able to export PROV information.

## 4.13 UV-CDAT Live

A client-server application, UV-CDAT Live gives scientists a mobile solution for performing big-data analytics and visualizations. The underlying architecture of UV-CDAT Live was specifically designed with the following scientific goals:

- To perform an integrated and scalable analysis of the high volume heterogeneous data collected from multiple disciplinary domains within the field of climate.
- To visualize the scientific findings with customized options that range from 1D plots to 3D graphics and animations.
- To share the scientific results in a reproducible and systematic manner through provenance.

Unlike UV-CDAT (the stand-alone version), UV-CDAT Live does not require any software installation other than the required web browser. By using UV-CDAT Live, scientists can begin their data analysis by simply opening a web browser, or by importing an existing project, which they have previously saved. In our current UV-CDAT Live design, there are 2 ways to import data sets from ESGF to UV-CDAT Live: by clicking on a link next to the desired data set or importing all of the selected data sets in the data cart. Currently, the caveats of UV-CDAT Live are the implementation of the underlying architecture that provides security and scalability and the cross-browser support at the front-end. However, both are to be resolved in a subsequent release (i.e., release version 2.0).

UV-CDAT Live adopts the Representational State Transfer (REST) architectural style, developed by W3C, that consists of clients and servers. A client sends a request to a server, and the server then processes the request and returns an appropriate response to the client. The server-side of UV-CDAT is powered by Django, which is a web framework that offers a object-relational mapper for database support, a cache system for scalability, security and session support, a flexible URL design without any framework-specific restrictions, and a template system that facilitates web content development. All server-side scripting is written in Python to interface with the existing UV-CDAT libraries, including CDMS, Numpy, VisTrails, etc., which are also written in Python. One major component of server-side is to provide data security. In UV-CDAT Live, all users are required to login at the beginning of a session. UV-CDAT Live uses ESGF's login account for authentication and authorization for data access, which is only given to those who are authorized as defined by their ESGF account. All usernames entered at the time of login are stored in a local database for tracking of user activity. Currently, OPeNDAP is used to deliver remote data to the Django server for data analysis and image rendering. For scalability, the future plan is to store data on a server that will delegate all computational tasks to a cluster and then deliver user-requested results to the Django server, which will then dynamically generate an HTML response with the user-requested contents sent back to the user.

The web client-side of UV-CDAT uses the template system provided by the Django to deliver dynamic contents to users. The templates are HTML pages, which retrieve data from a database as defined in the Python/Django settings. Currently, these templates are written in HTML5, CSS3, and JavaScript. Ajax and jQuery, which are various forms of JavaScript, are also being used. Once a user is authenticated and authorized, a main web page designed for data analysis online is returned to the user. As shown in Figure 40, the web page contains the following components:

- Data set panel that lists the desired data sets and their associated variables.
- Graphic spreadsheet to display plots.
- Variable panel that lists the desired variables.
- Geographical coordinate panel, including a Google map to select coordinates.
- Plot type selection panel to select the plot type.
- Calculator to setup equations to create new variables.

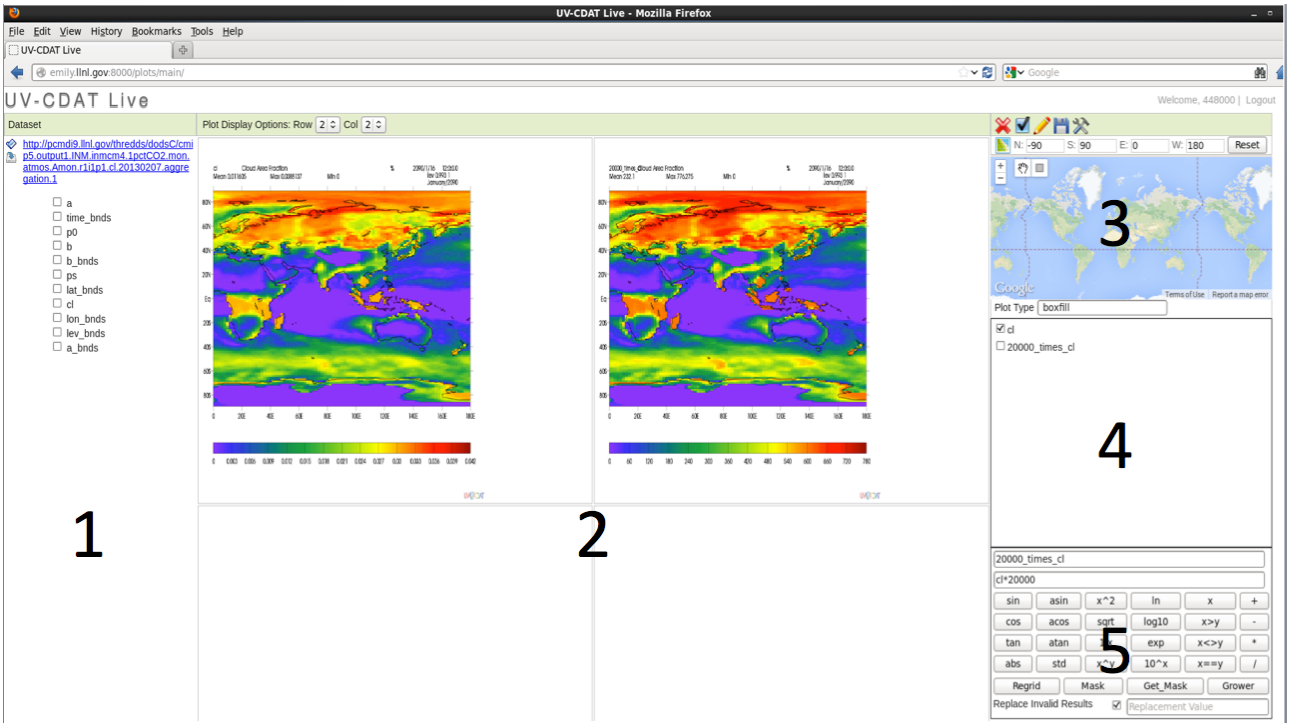



**Figure 39.** *The UV-CDAT Live client user interface is modeled after the UV-CDAT GUI client. Data are obtained via secured ESGF access channels. However, data are reduced and analyzed on the data's own server. If desired, the user can download the data to their desktop for further analysis or display the data in the graphic spreadsheet. As UV-CDAT's GUI, the Web UI is divided into many primary parts: the variable listed from within ESGF, visualization spreadsheet, Google map to select sub-regions,derived and ingested variables; and calculator.*

The following features have been implemented for UV-CDAT Live:

1. Security: A login page has been built for users to enter their username and password. The server will return an error message at the top of the page for any incorrect input of username and/or password, where the users can try to login again on the same page. Also, an access-denied page is returned to those users who do not have the authorization to access a specified data set.
2. Multiple User Mode: NetCDF was modified to allow multiple users to access UV-CDAT Live simultaneously.

3. Main Web Page:
    a. In the data set panel, there are two buttons associated with each file: an open/closed button to toggle the list of variables of the corresponding file, and an import button that will import all the selected variables to the variable panel. Clicking on the file will trigger a download of all the selected variables. If no variables were selected, then the entire file will be downloaded.
    b. In the variable panel, one can select all the variables by clicking on the checked-checkbox icon at the top of the variable panel. All the selected variables can be deleted by clicking on the "X" icon. Also, when clicking on the pencil icon, an input field will appear for all the selected variables for editing, and a user can simply press "return" when finished editing and the corresponding input field will disappear. In addition, there is a reset icon for users to uncheck all the checkboxes and remove all input fields in the variable panels.
    c. In the canvas spreadsheet, there are four slots to display graphics. The row and column dropdown lists located on the top of the spreadsheet allow users to change the size of the canvas spreadsheet. A user can simply drag a variable from either the data set panel or the variable panel into a canvas to render a plot. When dragging from the data set panel, the same variable will be automatically imported to the variable panel.
    d. In the geographical coordinate panel, a user can click on the map icon to drop down a Google map, in which a user can drag a mouse across the map to define a set of coordinates. The zoom option on the map is also enabled whereby the user can zoom in and out of a region. The default values, as shown in the input fields, are -90, 90, 0 and 180 for north, south, east and west, respectively. A reset button is implemented to revert the coordinates back to the default setting.
    e. The plot type panel is currently set as boxfill. Further development is in progress to add other plot types that are available in the UV-CDAT standalone version.
    f. A calculator was added to the web page to edit the values of a variable. This calculator includes an input field to define the name of the new variable and another for entering an equation. Currently, a multiplication button has been implemented. One can enter the name of a new variable, select the desire variable from the variable panel as the multiplier, click on the multiply button on the calculator, and then enter the value for the multiplicand. Once the return key is pressed, the new variable will be automatically be added to the list of variable in the variable panel.

# 5 Community Outreach

For UV-CDAT to be a success, it must become the de-facto open-source tool for visualization and analysis of big climate data across the globe. Widespread use will also benefit UV-CDAT technically, as it becomes part of other government and non-government projects.

In order to realize our admittedly high-flown goal, the UV-CDAT team has taken a series of steps in community outreach. One is to offer workshops and online webinars and classes for users and developers. The first international workshop will be offered in December 2013, as described further below. Another has been to provide an extensive collection of documentation and publications. We also electronically manage all libraries and packages using GitHub for manually registering and tracking bug reports and software updates. With our completely web-based system, adding bug reports and viewing fixes is as simple as filling in a few details on the UV-CDAT GitHub website and registering a bug report. For more information on UV-CDAT, email us at uvcdat-support@llnl.gov or visit our website at http://uv-cdat.org/.

## 5.1 Workshops

During the first week of December 2013, UV-CDAT, in conjunction with ESGF, will host its first international workshop. The focus of the workshop is to bring technologists and scientists interested in leveraging emerging technologies and solutions to support, enable, and enhance climate and weather research. Of particular interest are individuals who are building infrastructures and tools to support data sharing, analysis and sharing of climate model output, observations, and reanalysis. U.S. principal investigators from DOE and NASA will organize the workshop along with international principal investigators from the Infrastructure for the European Network for Earth System Modelling (IS-ENES 2) project. This partnership envisions an environment that allows users open access to exabytes of model-generated, satellite, and in-situ data including physical, biogeochemical and ecosystem content. This infrastructure will bring to life a synchronized federation that supports models, observations, reanalysis, GIS integration, model intercomparison capabilities, user support, and visualization, as well as a full suite of server-side analysis tools and integration with desktop and web-based productivity applications. More information on the workshop can be found at the UV-CDAT URL link.

Yearly UV-CDAT workshops will focus on both “user-centric” and “developer-centric” issues that reflect the dynamic needs of the broader community and help to shape future software development. Critical issues include *technical challenges*, such as developing and implementing open standards and interoperability between software components; *legal challenges*, such as terms-of-use, group policies, or trust frameworks and new paradigms; *usability challenges*, such as how application and user interfaces are created so that using and navigating different software components are made easy; *social challenges*, such as determining how technologies, tools and services will impact social interactions; *local hardware challenges*, such as supporting the many different operating systems on a host of flavored platforms; and *server and cloud challenges*, such as how to work with a whole range of sever-side computing platforms connected via a complex globally distributed heterogeneous network.

## 5.2 Documentation and Online Training

To support both the user and developer communities, the core UV-CDAT team—LLNL, NASA, ORNL, LANL, NYU-Poly, and Kitware—built upon the existing UV-CDAT website and used progressive open-source documentation and proven tutorial tools (i.e., Sphinx, Docutils, MaxMind, YouTube, and others) to deliver a robust user and developer support system.

Added to documentation and training are written text for articles, book chapters, and reports. See Appendix D for the lengthy list of outreach activities and publications. The UV-CDAT team continues to publish journal articles and book chapters about UV-CDAT and have several under peer-review. Publications can be found here. A formal written document on UV-CDAT use is in progress and will be included with the future release of UV-CDAT release version 2.0.

In addition, we will build an online library of tutorials by archiving a bi-monthly online webinars, for hands-on classes and remote learning. Our initial tutorials are designed to help users get started with UV-CDAT. Viewing online requires only a web browser and Internet connection. Community suggestions for additional tutorials may be entered on the UV-CDAT website. See also the Tutorial link to the first webinars at our website.

# 6 Collaborative Governance

Collaborative governance is defined by Wikipedia as a process and a form of governance in which participants (parties, agencies, stakeholders) representing different interests are collectively empowered to make a policy decision or make recommendations to a final decision-maker who will not substantially change consensus recommendations from the group. Collaborative governance has characteristics of both collaboration and empowerment.

For UV-CDAT, the governance direction is to take it from a benevolent Principal Investigator (PI) (or malevolent PI, depending on who you talk to) hierarchical governing body to a flat collaboratively governed body. This governance model will give UV-CDAT the longevity it needs to persist over time and the flexibility required to adjust to the community's short- and long-term needs. The new governing body, called the UV-CDAT Review Board (URB), will be a group of individuals whose goals are to advance UV-CDAT by providing direction and oversight to its development. The URB does not yet exist. We are awaiting formalization of the board by program mangers.

While the open-source nature of UV-CDAT allows natural progression via its many contributors and collaborations, the board will provide synergy and cohesion among the various developmental efforts, thus ensuring that change benefits the community as a whole. The URB will maintain a roadmap for UV-CDAT, including long-term plans, and make decisions on high-impact code changes to UV-CDAT.

The URB will include the following groups:

- A steering committee, composed of funding agencies and stakeholders responsible for providing resources.
- An executive committee, with overall responsibility for meeting sponsors, stakeholders, and community needs and prioritizing work.
- A technical committee, responsible for the development of the system architecture, the management of the development lifecycle, and scheduling releases.

The executive and technical committees are responsible for setting release contents and reporting what was actually delivered in the releases to the community.

Code changes with a high impact on developers and/or users are to be reviewed by the URB technical committee. Guiding principles for deciding whether a change requires URB involvement include:

- Will the change significantly affect backwards compatibility?
- Will the change significantly affect users, managers, or developers?
- Does the change significantly shift the functionality and scope of UV-CDAT?
- Are there any licensing issues?
- Are there any security concerns that need to be addressed?

Bug fixes will not in general require URB approval. Small feature additions also will not require approval provided they are part of an associated development plan. In both cases, changes must be formally documented and published to the mailing list and/or tracking web sites (i.e., UV-CDAT's GitHub site) for comment before implementation. Responsibilities and details on each URB committee are noted:

- ***Steering Committee (SC) Responsibility:*** Comprised of program managers whose agencies provide significant sources of identified and dedicated funding and resources for UV-CDAT development and core infrastructure. The SC provides a forum for UV-CDAT funding agencies to interact on a regular basis regarding the vision, goals, and progress of the UV-CDAT effort as they relate to each agency's mission and plans. Agency representatives work with the UV-CDAT executive and technical committees to articulate UV-CDAT goals and formulate metrics that reflect programmatic requirements. The executive committee

reports to the SC on the status of UV-CDAT development, adoption, and collaborations, and offers guidance as warranted. A program manager designated by the Climate and Environmental Sciences Division within the DOE Office of Science Biological and Environmental Research will initially **Chair** the SC. The **Chair**, in consultation with other agency representatives who support UV-CDAT, will determine the funding threshold for membership on the Steering Committee. Roles of the steering committee include:

- Communicating agency objectives and constraints to the executive committee to ensure that they are reflected in UV-CDAT goals and vision.
- Reviewing and concurring on UV-CDAT programmatic metrics proposed by the executive committee, and changes to those metrics as may be necessary from time to time.
- Advising on funding issues and opportunities that may be relevant to continued progress.
- Ensuring significant sponsoring agencies are represented on the SC.

- ***Executive Committee (EC) Responsibility:*** General guidance and overall high-level decisions in directing the course of the UV-CDAT project in correspondence with multiple sponsor and stakeholders needs—ultimate responsibility for ensuring that UV-CDAT meets the needs of customers and stakeholders. Future EC membership shall develop organically from the EC itself. EC members are responsible for nominating new members, who are elected by consensus or majority vote (with the **Chair** breaking any tie). Members may step down from the EC at any point. Members may be expelled from the EC by consensus or vote if they are unable to attend meetings after reasonable effort has been made to contact them, or if their actions are found to be counterproductive to the purposes of the URB. The executive committee is responsible for organizing and interacting with the UV-CDAT Steering, Technical and Advisory committees. The steering committee—composing of sponsors, stakeholders, and users—provides input to the EC on the entire UV-CDAT project and path forward from their perspective. This input will be interpreted and used to provide requirements to the technical committee. The executive committee **Chair** must attend all steering committee meetings. The **Chair**, when permitted, will attend all technical committee meetings to provide input. Roles of the executive committee include:
  - Setting the strategic direction of the UV-CDAT project.
  - Oversight of technical committee (and other committees as defined by the executive committee).
  - Work with technical committee to review implantation plan for strategic goals.
  - Approval of major architectural changes.
  - Appoint an advisory committee of users and external stakeholders who will provide feedback to the EC and technical committee on plans and suggestions for new directions.

- ***Technical Committee (TC) Responsibility:*** Guidance and direction on technical issues set by executive committee requirements—focused on technical aspects of UV-CDAT. Future TC membership shall develop organically from the TC itself. TC members are responsible for nominating new members who are elected by consensus or majority vote (with the **Chair or co-Chairs** breaking any tie). Technical members may step down from the TC at any point. Members may be expelled from the TC by consensus or vote if they are unable to attend meetings after reasonable effort has been made to contact them or if their actions are found to be counterproductive to the purposes of the URB. The **Chair** or **co-Chairs** will attend all executive and technical committee meetings to provide input. Roles of the technical committee include:
  - Requirements analysis (interfaces with customers - users - to understand requirements).
  - Technical architecture.
  - Organizing development.
  - Testing and releases.

The full UV-CDAT governance document can be seen on the UV-CDAT website or at: http://aims.llnl.gov/uvcdat-site/governance.html.

# 7 The Future of UV-CDAT

To support the development, execution and validation of DOE's next-generation ESM, the team will extend UV-CDAT's knowledge discovery through a web-based user interface that will incorporate diagnostic packages with automated analysis for well known climate features and predefined regions of interests; knowledge sharing through collaborative technologies that records, stores, and builds on user discoveries; heterogeneous computation framework that builds in parallel analysis packages to support intensive computational needs; and state-of-the-art visualizations for the ever growing needs of the climate science community. By leveraging UV-CDAT capabilities and other supported DOE projects, such as ESGF and Globus Online, the larger DOE team will develop an end-to-end data test bed to meet the DOE's ambitious goals for sharing, movement, management, and analysis of big distributed data.

Many challenges remain for the UV-CDAT team, and new ones will emerge as more diverse data sets become available from climate simulations run on ever-larger HPC platforms with data from increasingly powerful, higher-resolution satellites and instruments. The team envisions an analytic and informatics infrastructure that provides even greater user support, community outreach, diagnostic packages, big data and parallel data processing, web- and server-side informatics and visualization, improved scripting capabilities, and more analytics and visualizations.

In the near-term, the team will focus efforts to expand into new areas of data mining, parallelism, workflows, HPC, cluster computing, and analysis over a distributed network. Furthermore, we intend to fully explore the possibility of providing a configurable and scalable cloud-based environment, so that resources and services can be instantiated on demand for specific short-lifetime projects or to meet such requirements as elastic allocation of computing processes. Some of the most prominent future activities include:

- *User and developer support:* Provide new documentation, tutorials, and website updates, maintenance, support for platforms, monitoring of system tools usage, etc. Engage with the community for requirements, adoption, and feedback.
- *Community outreach:* Engage with the climate-science community in the form or workshops, governance, literature, etc.
- *Diagnostic packages:* Develop AMWG, LMWG, OMWG, PCWG, and Model for Prediction Across Scales (MPAS) diagnostic packages along with others to be determined.
- *Big data/parallel data processing architecture:* Provide three parallel processes that have been identified for running batch and interaction parallel processes along with client-server parallel interaction.
- *Web and information visualization:* Advance UV-CDAT analysis and visualization capabilities to a web application.
- *Software process:* Continue to build upon and improve UV-CDAT's build, test, and documentation processes.
- *Refactor code to make it easier to add functionality:* Simplify the extendable and pluggable feature to support the addition of new codes.
- *Interface enhancements:* Partner with the climate science community to improve the user experience.
- *Improve scripting:* Improve workflow scripts and user-controlled parameters.
- *Improve visualization:* Develop new 2D and 3D visualization and interface for the BER and NASA climate community.

For the DOE ESM project, there must be a package that will run automated and interactive diagnostics on the output. UV-CDAT scripts are being developed to generate the data files required to produce the 600+ AMWG diagnostics and plots. Currently, over 100 AMWG diagnostics and plots can be created using UV-CDAT AMWG

diagnostics package. In place of the existing NCAR Command Language (NCL)-based AMWG package with static plots, the UV-CDAT team envisions dynamic user interaction and query through the UV-CDAT interface to allow functions such as changing contour intervals and colors, zoom to regional refinement, and displaying multiple user selected plots on one canvas. Over the longer term, the diagnostics will be expanded to cover not only the AMWG but also the ocean, land, and polar climate model working groups in FY 14.

*The UV-CDAT team envisions their software as a game-changer in the analytics/informatics and visualization sciences—an infrastructure that provides a one-stop shop for big data mining through a parallel interface to a rich set of heterogeneous, large-scale data holdings from diverse science domains.*

# 8 References

[CF, 2009]: CF, Climate and Forecast (CF) metadata convention for processing and sharing data files (2009). http://cf-pcmdi.llnl.gov/.

[Chacon, 2013]: Pro Git book, written by Scott Chacon and published by Apress, 2013. http://git-scm.com/book/.

[CMOR, 2009]: CMOR, the Climate Model Output Rewriter produces CF-compliant netCDF data files (2009). http://www2-pcmdi.llnl.gov/cmor.

[Doutriaux, 2007]: Doutriaux, Charles, Drach, Robert, Williams, Dean, 2007: The Visualization and Control System (VCS), http://www2-pcmdi.llnl.gov/cdat/first-page/cdoutrix/beginners-guide/data-visualization/?searchterm=VCS.

[Drach, 2007]: Drach, Robert, Dubois, Paul, and Williams, Dean, 2007: Climate Data Management System, version 5.0, http://www2-pcmdi.llnl.gov/cdat/manuals/cdms5.pdf.

[HDF Group, 2009]: HDF Group, HDF: a file format used for storing various forms of data (e.g., raster, scientific, etc.) (2009). http://www.hdfgroup.org/.

[ParaView, 2013]: ParaView visualization reference. http://pvw.kitware.com/.

[PP, 2013]: PP-format is a record-based binary format used in a number of the BADC's data sets. It is a Met Office proprietary format so is mainly associated with Met Office products. http://badc.nerc.ac.uk/help/formats/pp-format/index.html.

[Sargsyan, 2013]: Khachik Sargsyan, Cosmin Safta, Habib N. Najm, Bert J. Debusschere, Daniel Ricciuto, Peter Thornton, "Dimensionality Reduction for Complex Models via Bayesian Compressive Sensing", accepted for publication, International Journal for Uncertainty Quantification, 2013.

[Taylor, 2011]: Karl E. Taylor, Ronald J. Stouffer, Gerald A. Meehl, "An Overview of CMIP5 and the Experiment Design", *Bull. Amer. Meteor. Soc.*, **93**, 485–498. doi: http://dx.doi.org/10.1175/BAMS-D-11-00094.1

[UCAR, 2008]: UCAR, University Corporation for Atmospheric Research, netCDF: a machine-independent, self-describing, binary data format (2008). http://www.unidata.ucar.edu/software/netcdf.

[WMO, 2009]: WMO, World Meteorological Organization standardized the GRIdded Binary (GRIB) data format commonly used in meterology to store historical and forecast weather data (2009). http://www.grib.us/.

[XDATA, 2013] XDATA application software. http://xdata.kitware.com/.

# Appendix A UV-CDAT Consortium and Task Summary

Here we describe the tasks of each institution that allowed us to meet both the research and science requirements and technical challenges and development documented in this report. The following is a summary of institutional primary tasks:

- *DOE national laboratories:* UV-CDAT's integrated cross-institutional effort required unique breadth and depth of expertise. The four DOE national laboratories—Lawrence Berkeley (LBNL), Lawrence Livermore (LLNL), Los Alamos (LANL), and Oak Ridge (ORNL)—focused on the development of large-scale parallel analytics and the diagnosis of climate model simulation (atmosphere, land, ocean, and sea-ice) and observational post-processing.
- *The two universities:* Polytechnic Institute of New York University and the University of Utah were responsible for provenance and workflow and streaming visualization.
- *Private sector:* Kitware and Tech-X worked on cross-platform build and test suites and supplied regridding and reprojections libraries.
- NASA's Goddard Space Flight Center provided 3D data visualization and UV-CDAT tutorials.

Task summary of institutional leadership and participation over the past three years.

● Task lead ■ Task Participant △ Task co-leads

| TASKS | LANL | LLNL | ORNL | LBNL | Univ. of Utah | NYU-Poly | NASA | Kitware | Tech-X |
|---|---|---|---|---|---|---|---|---|---|
| UV-CDAT overall software integration | ■ | △ | ■ | ■ | ■ | ■ | ■ | △ | ■ |
| Parallel analysis and I/O mechanisms for large scale visualization and analysis | △ | △ | △ | △ | | | | △ | △ |
| Develop mechanisms for regridding, reprojections, and aggregation for observational datasets | ■ | ■ | ■ | | | | | | ● |
| Develop mechanisms for regridding, reprojections, and aggregation for model simulation datasets | ■ | ● | ■ | | | | | | ■ |
| Develop mechanisms for regridding, reprojections, and aggregation for observation datasets | | | △ | | | | △ | | ■ |
| Develop mechanisms for ensemble analysis | ■ | ■ | ● | | | | | | |
| Spatiotemperal data mining architecture | △ | △ | △ | | | | | | |
| Workflow and Provenance Architecture | | | | | | ● | | | |
| UV-CDAT tightly coupled architecture | ■ | ■ | | | | △ | ■ | ■ | ■ |
| UV-CDAT loosely coupled architecture | | | ■ | ■ | ■ | △ | | | |
| Develop cross-cutting use-cases for each of the projects subsections (models and reanalysis simulations, and satellite and instrument observations) | △ | △ | △ | △ | | | △ | | |
| Streaming data formats | | △ | | | △ | | | | |
| Topological analysis techniques | ■ | ■ | ● | ■ | | | ■ | | |
| Model (atmosphere, land, ocean) diagnostics tools | △ | △ | △ | | | | △ | | |
| Comparative visualization and analysis | △ | △ | △ | △ | | | △ | △ | |
| Extend the ESGF metadata schema and the federated metadata catalog for analyzing datasets | ■ | ■ | ■ | ■ | | ● | | | |
| Use interface to ESGF from UV-CDAT | | ● | | | | ■ | ■ | | |
| Cross-platform build and test suite | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ● | |

# Appendix B Milestones and Software Releases

UV-CDAT software releases over the past years.

| Product | Release Date | End of Engineering | End of Life/ End of Support |
|---|---|---|---|
| UV-CDAT 2.0.0 | March 1, 2014 | Open | |
| UV-CDAT 1.4.0 | September 25, 2013 | January 4, 2013 | January 4, 2013 |
| UV-CDAT 1.3.0 | April 29, 2013 | August 5, 2013 | September 18, 2013 |
| UV-CDAT 1.2.0 | January 6, 2013 | August 5, 2013 | April 29, 2013 |
| UV-CDAT 1.1.0 | June 29, 2012 | April 29, 2013 | January 6, 2013 |
| UV-CDAT 1.0.0 | May 9, 2012 | January 6, 2013 | June 29, 2012 |
| UV-CDAT Beta | April 4, 2012 | June 29, 2013 | May 9, 2012 |
| UV-CDAT Alpha | February 2, 2012 | May 9, 2012 | April 4, 2012 |

# Appendix C Collaborations

From the beginning, collaboration has been the UV-CDAT watchword, enabling the team to make a major impact on the progress of science in IPCC AR5, the DOE-Earth System Modeling project, and the European IS-ENES G8 project. We expect to continue to provide expertise to geoscience projects in need of infrastructure for analytics and informatics.

One ongoing collaborations is NASA's **I**ntegrated **C**limate **A**nalysis Dist**r**ib**u**ted **S**ervices (ICARUS). ICARUS' main goal is to facilitate the joint discovery and analysis of NASA satellite observations and climate model output. As reported in the literature, satellite observations are a key to understanding why long- and medium-term predictions from different models diverge, even when the models are run with the same assumptions about future human development and anthropogenic emissions. ICARUS will remove current roadblocks by providing an integrated platform where models and observations are accessed through the same services and with the same semantics. ICARUS will specifically target the community of thousands of users that have already enrolled for access to CMIP5 model output stored and served by the ESGF. By addressing this existing community, we will establish a state of practice that will carry on for future CMIPs and IPCC reports, thus guaranteeing an ever-increasing relevance of NASA observations for DOE climate change research.

The team has also had discussions with groups proposing analytic data centers and institutes for which the UV-CDAT technology is highly relevant. In most cases, these projects are developing tools and technologies that would be useful to UV-CDAT, and in some cases there is an interest in generalizing and enhancing UV-CDAT-developed technology and disseminating it to a larger audience.

The National Science Foundation (NSF) is considering using the UV-CDAT environment to integrate NSF's scientific computing libraries and applications to foster more efficient use and analytics. UV-CDAT technology may be used as:

- An EarthCube, in which DOE and NSF share a joint vision for large-scale analysis and visualization for both observational and model-generated climate data, with the aim of delivering new geoscience capabilities into the hands of scientists to foster more efficient knowledge discovery.
- A NSF ClimateTranslator where, through an application programming interface, a flexible translator assists in analysis, evaluation, synthesis, and interpretation while capturing provenance and metadata to ensure reproducibility.
- Software infrastructure, for which UV-CDAT's pluggable framework of open-source analytics and computation services and tools would be deployable in a cloud environment to provide functionality essential for broad range "big data" scientific computing.

# Appendix D Outreach, Publications, etc.

To effectively build an infrastructure capable of dealing with large-scale data analysis and visualization, we established connections with other funded DOE Office of Science projects and national and international programs and agencies at various meetings, workshops, and conferences. This section describes many of the substantive interactions we have had during the short life of the UV-CDAT funded project.

## *B.1 Outreach Activities*

- Charles Doutriaux and Dean Williams will host the UV-CDAT Face-to-Face Meeting to be held at LLNL in December 2013. This is a joint collaborative meeting between DOE, NASA, and the European IS-ENES G8 project. The focus of this meeting will be to bring together technologists and scientists interested in leveraging emerging technologies and solutions to support, enable and enhance climate and weather research. Of particular interest are individuals who are building infrastructures and tools to support data sharing, analysis of climate model output, observations, and reanalysis.
- Dean Williams hosted the LLNL, PMEL, and NOAA Development Meeting to join UV-CDAT and NOAA's Ferret analysis and visualization activities. The meeting was held at Lawrence Livermore National Laboratory in Livermore, California, March 2011.

## *B.2 Papers*

### *B.2.1 October 1, 2012, through September 30, 2013*

- Boonthanome Nouanesengsy, John Patchett, James Ahrens, Andrew Bauer, Aashish Chaudhary, Berk Geveci, Ross Miller, Galen M. Shipman, and Dean N. Williams. "Optimizing File Access Patterns through the Spatio-Temporal Pipeline for Parallel Visualization and Analysis". Submitted to LDAV 2013.
- Dean N. Williams, Gavin Bell, Charles Doutriaux, Andrew C. Bauer, Aashish Chaudhary, Harinarayan Krishnan, Michael Wehner, Boonthanome Nouanesengsy, John Patchett, Gerald L. Potter, Thomas P. Maxwell, Wei-Chen Chen, George Ostrouchov, Dave Pugmire, Galen M Shipman, Brian E. Smith, Chad A. Steed, Alexander Pletzer, "Climate Science 'Big Data' Parallel Analysis: Analyzing Mult-Model Climate Simulation Data", IEEE Parallel and Distributed Systems, under peer-review, due out in 2013.
- Chad A. Steed, Daniel M. Ricciuto, Galen Shipman, Brian Smith, Peter E. Thornton, Dali Wang, Xiaoying Shi, Dean N. Williams, "Big Data Analytics for Earth System Simulation Exploratory Analysis", Computers & Geosciences, under peer-review, due out in 2013.
- Luca Cinquini, Daniel Crichton, Chris Mattmann, Gavin M. Bell, Charles Doutriaux, Bob Drach, Dean Williams, John Harney, Galen Shipman, Feiyi Wang, Philip Kershaw, Stephen Pascoe, Rachana Ananthakrishnan, Neill Miller, Estanislao Gonzalez, Sebastian Denvil, Mark Morgan, Sandro Fiore, Zed Pobre, Roland Schweitzer, "The Earth System Grid Federation: An Open Infrastructure for Access to Distributed Geospatial Data", IEEE special issue of FGCS (Future Generation Computing Systems), under peer-review, due out in 2013.
- Dean Williams, "Dealing with Data Overload in the Scientific Realm", Science & Technology Review, January/February 2013, https://str.llnl.gov/january-2013/williams, https://str.llnl.gov/january-2013.
- Benjamin D. Santer, Jeffrey F. Painter, Carl A. Mears, Charles Doutriaux, Peter Caldwell, Julie M. Arblaster, Philip J. Cameron-Smith, Nathan P. Gillett, Peter J. Gleckler, John Lanzante, Judith Perlwitz, Susan Solomon, Peter A. Stott, Karl E. Taylor, Laurent Terray, Peter W. Thorne, Michael F. Wehner, Frank J. Wentz, Tom M. L. Wigley, Laura J. Wilcox, and Cheng-Zhi Zou, "Identifying Human Influences on Atmospheric Temperature", Proceedings of the National Academy of Sciences (PNAS) 2013 110 (1) 26-33; January 2, 2013, doi:10.1073/pnas.1210514109.

- Brian Smith, Daniel M. Ricciuto, Peter E. Thornton, Galen Shipman, Chad Steed, and Dean Williams. ParCAT: Parallel Climate Analysis Toolkit. In Proceedings of the International Conference on Computational Science, pp. 2367–2375, June 2013. doi:10.1016/j.procs.2013.05.408
- Dean N. Williams, Timo Bremer, Charles Doutriaux, John Patchett, Galen Shipman, Blake Haugen, Ross Miller, Brian Smith, Chad Steed, E. Wes Bethel, Hank Childs, Harinarayan Krishnan, Michael Wehner, Claudio T. Silva, Emanuele Santos, David Koop, Tommy Ellqvist, Huy T. Vo, Jorge Poco, Berk Geveci, Aashish Chaudhary, Andrew Bauer, Alexander Pletzer, Dave Kindig, Gerald L. Potter, Thomas P. Maxwell, "The Ultra-scale Visualization Climate Data Analysis Tools: Data Analysis and Visualization for Geoscience Data," *Computer*, 27 March 2013. IEEE Special Issue: Cutting-Edge Research in Visualization. IEEE computer Society Digital Library. IEEE Computer Society, http://doi.ieeecomputersociety.org/10.1109/MC.2013.119.
- Dean N. Williams, Ian T. Foster, Bryan Lawrence, Michael Lautenschlager, Don E. Middleton, "Earth System Grid Federation: Infrastructure to Support Climate Science Analysis as an International Collaboration", Chapter in Data Intensive Science: Critchlow, Terence and Kleese-Van Dam, Kerstin, Editors. Chapman & Hall/CRC Computational Science, 2013, http://www.crcpress.com/product/isbn/9781439881392.
- Brian Summa, Attilay Gyulassy, Peer-Timo Bremer, and Valerio Pascucci, "Interactive Data Exploration" Chapter in Data Intensive Science: Terence Critchlow, and Kerstin Kleese van Dam, Editors. Chapman & Hall/Crc Computational Science, 2013, http://www.crcpress.com/product/isbn/9781439881392.
- Brian Smith, Daniel M Ricciuto, Peter E Thornton, Galen Shipman, Chad Steed, Dean Williams, and Michael Wehner 2013 International Conference on Computational Science, Fourth Workshop on Data Mining in Earth System Science (DMESS) 2013.
- Emanuele Santos, Jorge Poco, Yaxing Wei, Shishi Liu, Bob Cook, Dean N. Williams, and Claudio T. Silva, "UV-CDAT: Analyzing Climate Data sets from a User's Perspecive", IEEE Computing in Science and Engineering: Visualization Corner, vol 15 (1), pp. 94-103, January 1, 2013. ISBN No. 1521-9615. DOI: http://doi.ieeecomputersociety.org/10.1109/MCSE.2013.15.

### *B.2.2 October 1, 2011, through September 30, 2012*

- Thomas P. Maxwell, "*Exploratory Climate Data Visualization and Analysis Using DV3D and UV-CDAT*", IEEE Computer Society, SCC '12 Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, pages, 483-487, 2012, ISBN: 978-0-7695-4956-9 doi>10.1109/SC.Companion.2012.69. http://www.nas.nasa.gov/SC12/demos/demo21.html
- S. Kumar, V. Vishwanath, P. Carns, J.A. Levine, R. Latham, G. Scorzelli, H. Kolla, R. Grout, R. Ross, M.E. Papka, J. Chen, V. Pascucci. "*Efficient data restructuring and aggregation for I/O acceleration in PIDX*", In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC12), IEEE Computer Society Press, pp. 50:1--50:11. 2012.
- Shusen Liu; Levine, J.A.; Bremer, P.; Pascucci, V., "*Gaussian mixture model based volume visualization*", Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on , vol., no., pp.73-77, 14-15 Oct. 2012. BEST PAPER AWARD.
- Wong, Pak Chung; Shen, Han-Wei; Pascucci, Valerio, Editors of special issue on "*Extreme-Scale Visual Analytics*", IEEE Transactions on Computer Graphics and Applications, vol.32, no.4, July-Aug. 2012.
- V. Pascucci, G. Scorzelli, B. Summa, P.-T. Bremer, A. Gyulassy, C. Christensen, S. Philip, S. Kumar. "*The ViSUS Visualization Framework*", In High Performance Visualization: Enabling Extreme-Scale Scientific Insight, Chapman & Hall/CRC Computational Science, Ch. 19, Edited by E. Wes Bethel; Hank Childs,

Lawrence Berkeley National Laboratory, Berkeley, California, USA; Charles Hansen, University of Utah, Salt Lake City, USA, Chapman and Hall/CRC, 2012.

- Gyulassy, A.; Kotava, N.; Kim, M.; Hansen, C.D.; Hagen, H.; Pascucci, V., "*Direct Feature Visualization Using Morse-Smale Complexes*", Visualization and Computer Graphics, IEEE Transactions on , vol.18, no.9, pp.1549-1562, Sept. 2012.
- Harsh Bhatia, Shreeraj Jadhav, Peer-Timo Bremer, Guoning Chen, Joshua A. Levine, Luis Gustavo Nonato, and Valerio Pascucci. Flow Visualization with Quantified Spatial and Temporal Errors using Edge Maps. IEEE Transactions on Visualization and Computer Graphics (TVCG), vol. 18, no. 9, pp. 1383-1396, Sept. 2012.
- Widanagamaachchi, W.; Christensen, C.; Bremer, P.-T.; Pascucci, V., "*Interactive exploration of large-scale time-varying data using dynamic tracking graphs*", Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on , vol., no., pp.9-17, 14-15 Oct. 2012. BEST PAPER AWARD FINALIST.
- Tierny, J.; Pascucci, V., "*Generalized Topological Simplification of Scalar Fields on Surfaces*", Visualization and Computer Graphics, IEEE Transactions on Visualization and Computer Graphics, vol.18, no.12, pp.2005-2013, Dec. 2012.
- E. Santos, D. Koop, T. Maxwell, C. Doutriaux, T. Ellqvist, G. Potter, J. Freire, D. N. Williams, and C. T. Silva, "Designing a provenance-based climate data analysis application." In Provenance and Annotation of Data and Processes, vol. 7525 of Lecture Notes in Computer Science, Springer, 2012.
- Chad A. Steed, Galen Shipman, Peter Thornton, Daniel Ricciuto, David Erickson, and Marcia Branstetter, "Practical Application of Parallel Coordinates for Climate Model Analysis," In Proceedings of the International Conference on Computer Science-Workshop on Data Mining in Earth System Science, June 2012, pp. 877-886. DOI: 10.1016/j.procs.2012.04.094.
- Williams, S., Petersen, M., Hecht, M., Maltrud, M., Patchett, J., Ahrens, J. and Hamann, B. (2012), "Interface Exchange as an Indicator for Eddy Heat Transport", Computer Graphics Forum, 31: 1125–1134. doi: 10.1111/j.1467-8659.2012.03105.x.

### *B.2.3 October 1, 2010, through September 30, 2011*

- Williams, S., Hecht, M., Petersen, M., Strelitz, R., Maltrud, M., Ahrens, J., Hlawitschka, M. and Hamann, B. (2011), "Visualization and Analysis of Eddies in a Global Ocean Simulation," Computer Graphics Forum, 30: 991-1000. doi: 10.1111/j.1467-8659.2011.01948.x
- Williams, S.; Petersen, M.; Bremer, P.-T.; Hecht, M.; Pascucci, V.; Ahrens, J.; Hlawitschka, M.; Hamann, B.; , "Adaptive Extraction and Quantification of Geophysical Vortices," Visualization and Computer Graphics, IEEE Transactions on , vol.17, no.12, pp.2088-2095, Dec. 2011oi: 10.1109/TVCG.2011.162
- B.D. Santer, C. Mears, C. Doutriaux, P.J. Gleckler, S. Solomon, T.M.L. Wigley, N.P. Gillett, D. Ivanova, T.R. Karl, J.R. Lanzante, G.A. Meehl, P.A. Stott, K.E. Taylor, P.W. Thorne, M.F. Wehner, and F.J. Wentz, "Separating Signal and Noise in Atmospheric Temperature Changes: The Importance of Timescale", *Journal of Geophysical Research* (Atmospheres), vol. 116, issue D22, November 2011. DOI: 10.1029/2011JD016263.

## *B.3 Presentations*

- Dean N. Williams presented, "Building a High-Performance Data and Compute Testbed Infrastructure for Extreme-Scale Climate Science" at the "Smoky Mountains Computational Science and Engineering Conference", held at the Park Vista hotel in Gatlinburg, TN, September 2013.

- Dean N. Williams and Galen Shipman presented, "Data Management and Analysis in Support of DOE Climate Science" at the Oak Ridge Leadership Computing: Processing and Analysis of Very Large Data Sets workshop, held at the Hilton hotel in Knoxville, TN, August 2013.
- Gerald Potter and Tom Maxwell presented UV-CDAT at the meeting titled, "*An Update on Obs4MIPs and Ana4MIPs: Part 2*", held at Langley Research Center in Hampton VA, June 2013.
- Gerald Potter and Tom Maxwell presented UV-CDAT at the meeting titled, "*An Update on Obs4MIPs and Ana4MIPs: Part 1*", held at Langley Research Center in Hampton VA, May 2013.
- Dean Williams and Claudio Silva presented and represented UV-CDAT at the Center for Urban Science + Progress (CUSP) Data Reproducibility Meeting held in Brooklyn, NY at NYU-Poly, May 2013
- Tom Maxwell and Gerald Potter presented UV-CDAT at numerous NASA meetings, on behalf of the development team. These NASA Goddard Space Flight Center (GSFC) presentations were held at:
    - The NASA Center for Climate Simulation (NCCS), March 2013
    - The GSFC/ Goddard Earth Sciences-Data and Information Service Center (GES-DISC), March 2013
    - The National Center for Environmental Prediction (NCEP), March 2013
    - NASA GSFC, December 2012
    - The NASA/Global Modeling Assimilation Office (GMAO), December 2012
    - NCEP, December 2012
- Dean Williams presented UV-CDAT at the Statistical and Applied Mathematical Sciences Institute (SAMSI)/NCAR Workshop on Massive Data sets in Environment and Climate Science, held at the Mesa Lab at NCAR in Boulder CO, February 2013.
- Charles Doutriaux, Tom Maxwell, Gerald Potter, and Dean Williams presented UV-CDAT on three separate occasions at the AGU Fall Meeting held in San Francisco CA, December 2012.
- Dean Williams and Robin Goldstone presented UV-CDAT at the 2012 BER Energy Sciences Network (ESnet) Requirements Review held in the Washington D.C. area, November 2012.
- Charles Doutriaux, Gerald Potter, and Dean Williams presented UV-CDAT at the 2nd Annual Earth System Grid Federation (ESGF) Conference held at Lawrence Livermore National Laboratory in Livermore CA, November 2012.
- Dean Williams presented UV-CDAT and ESGF as an invited speaker to Argonne National Laboratory, in Argonne IL, November 2012.
- Gerald Potter presented ESGF and UV-CDAT, as an update on Obs4MIPs and Ana4MIPs, at the "*Sounder Science Team Meeting*", Greenbelt MD, November 2012.
- Dean Williams presented UV-CDAT working within the CSSEF testbed at the CSSEF Face-to-Face Meeting in the Washington D.C. area, November 2012.
- Dean Williams presented the "*Data Analysis and Visualization: UV-CDAT*" developments at the G8 Initiative ExArch (IS-ENES-2) Kickoff meeting. The meeting was held in London, UK at the Royal Adelaide Hotel, October 2012.
- Gerald Potter presented UV-CDAT and DV3D at the "*Clouds and the Earth's Radiant Energy System (CERES) Science Team Meeting*", held at the Geophysical Fluid Dynamics Laboratory (GFDL) in Princeton NJ, October 10, 2012.

- Dean Williams presented UV-CDAT at the 2012 Second International Workshop on "*Climate Informatics*" held at the NCAR's Institute for Mathematics Applied to Geosciences in Boulder, Colorado, September 2012. At the workshop, members of the organization committee from universities expressed interested in collaborating with in the development of UV-CDAT. In particular, Cal Tech was interested in including their observational diagnostics into UV-CDAT.
- Dean Williams presented the "*Climate Analytics, Informatics, and Management Systems: BER Data Projects, Software, and Tools*" on behalf of the community. The meeting was held at the Gaithersburg Marriott Washington Center in Gaithersburg, MD, September 2012. UV-CDAT was presented alongside other BER software supported projects, such as CMOR, ESGF, testbeds, etc.
- Dean Williams presented ESGF/UV-CDAT at the Institute for Computing in Science (ICIS) "Big Data and Long Tails" Workshop, held in Park City Utah, August 2012.
- Chad A. Steed, "*Visual Analytics for Climate Model Analysis*", invited presentation to the Naval Research Laboratory, Stennis Space Center, MS. July 2012.
- Dean Williams presented ESGF/UV-CDAT at the DOE BER Climate and Environmental Sciences Division (CESD) and DOE ASCR Data Intensive meeting discussions held in Germantown, MD, June 2012.
- Chad A. Steed, "*Visual Analytics for Climate Analysis*", invited presentation to the Tennessee Valley Authority (TVA) River Forecast Center, Knoxville, TN, May 2012.
- Chad A. Steed, "*Visual Analytics for Climate and Text Analysis*" invited seminar presentation to the University of Tennessee Center for Intelligent Systems and Machine Learning (CISML), Knoxville, TN. April 2012.
- Chad A. Steed, "*Visual Analytics for Climate and Text Analysis*", invited seminar presentation to the Mississippi State University Computer Science and Engineering Department, Starkville, MS. April 2012.
- Dean Williams presented ESGF/UV-CDAT at the Earth System Prediction Capability (ESPC) workshop held at the NOAA Science Center in Silver Spring, MD, March 2012.
- Chad A. Steed, "*Visual Analytics Tools for CLM*", workshop presentation to the Joint Land and Biogeochemistry Working Groups, NCAR Mesa Lab, Boulder, CO, March 2012.
- Dean Williams presented UV-CDAT at the larger CSSEF Workshop meeting in Washington, D.C. held February 2012.
- Dean Williams presented and represented UV-CDAT at the LLNL General Electric collaboration meetings held February through June 2012.
- Dean Williams presented UV-CDAT at the CSSEF Face-to-Face Meeting held in the Washington D.C. area, January 2011.
- Dean Williams presented UV-CDAT at the Scientific Collaborations for Extreme-Scale Science Workshop, held at the Gaithersburg Marriott Washington Center in Gaithersburg, MD, December 2011.
- Dean Williams presented UV-CDAT at the 2011 AGU Fall Meeting (at the DOE BER Town Hall meeting) held in San Francisco, CA, December 2011.
- Dean Williams presented UV-CDAT at the DOE Climate Model Meeting, in Washington D.C. area, September 2011.
- Dean Williams presented UV-CDAT at the Climate Science for a Sustainable Energy Future (CSSEF) Kick-off Meeting, held in Gaithersburg, MD, August 2011.
- Dean Williams presented UV-CDAT at the Global/Climate Change Portal Requirements and Implementation Discussion Webinar, July 2011.

- Dean Williams presented UV-CDAT at the 9th Global Organization for Earth System Science Portals (GO-ESSP) Workshop, held in NOAA's National Climatic Data Center (NCDC) in Asheville, NC, May 2011.
- Dean Williams presented UV-CDAT and ESGF at the Exascale Data Management, Analysis, and Visualization Workshop, held in Houston TX, February 2011.
- Dean Williams presented UV-CDAT at the Conference on Interactive Information Processing Systems (IISP), 91st AMS Annual Meeting, held in Seattle Washington, January 2011.
- Dean Williams presented UV-CDAT at the "*Pacific Northwest National Laboratory and LLNL Meeting*" held in Richland, WA, December 2010,
- Dean Williams presented UV-CDAT at the "*NASA Technical Data Systems Workshop*" held at the Goddard Space Flight Center in the Washington, D.C. area, November 2010.
- Dean Williams presented UV-CDAT at the "*Greenhouse Gas Information System (GHGIS) Mission Operations Center (CMOC)*" held in Livermore, CA, October 2010.

### B.4 Posters

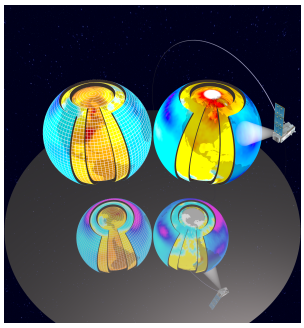
- UV-CDAT Team, "*Recent Achievements in Visual Data Exploration and Analysis for Climate Science*", at the DOE Scalable Data Management, Analysis, and Visualization Workshop, 2012.
- Jerry Potter, Dean Williams, and Laura Carriere, "*Making Reanalysis Data Available Through the Earth System Grid Federation*", at the Fourth WCRP International Conference on Reanalysis, Silver Spring, MD, May 2012.
- Tom Maxwell, "*Ultrascale Climate Data Visualization and Analysis*", at the 2012 Supercomputer Conference, held in Salt Lake City, Utah, November 2012.
- Allen Kou, Charles Doutriaux, and Dean N. Williams presented the poster, "*Qt Development for Ultra-scale Visualization Climate Data Analysis Tools*" at the annual LLNL Research Poster Session, held in Livermore, California in August 2010.

### B.5 Reports

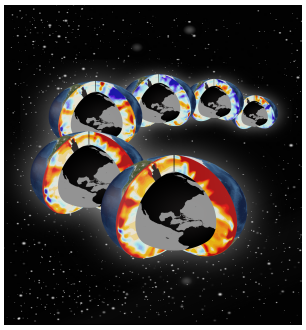
- Report from the Center for Urban Science + Progress (CUSP) Workshop on "*Software Infrastructure for Reproducibility in Science*", due to be released in 2013.
- Report from the DOE ASCAC Data *Subcommittee on "Synergistic Challenges in Data-Intensive Science and Exascale Computing*", released March 2013.
- Report from the Biological Environmental Research Network Requirements on "*BER Network Requirements Review Final Report*", conducted November 2012.
- Report from the DOE ASCR 2011 Workshop on "*Exascale Data Management, Analysis, and Visualization*", conducted February 2011.

### B.6 Animations

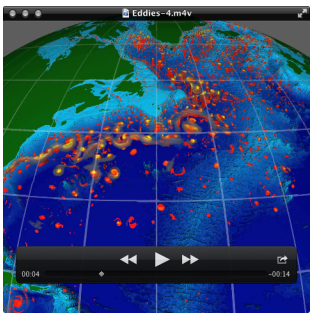
- Ben Santer et al., Proceedings of the National Academy of Sciences (PNAS) 2013, "*Identifying human influences on atmospheric temperature*", animation on multi-model detection and attribution study with climate model simulation output and satellite based measurements of tropospheric and stratospheric temperature change, June 2012. [http://aims.llnl.gov/uvcdat-site/animations.html]



- Peter Gleckler et al., Nature Climate Change | Letter, "*Human-induced Global Ocean Warming on Mutildecadal Timescales*", animation on increases in upper-ocean temperatures. Several studies have used well-established detection and attribution methods to demonstrate that the observed basin-scale temperature changes are consistent with model responses to anthropogenic forcing and inconsistent with model-based estimates of natural variability, June 2011. [http://aims.llnl.gov/uvcdat-site/animations.html]



- Sean Williams et al., "*Visualization and Analysis of Eddies in a Global Ocean Simulation*", The Okubo-Weiss parameter visualized at depth by removing all low-vorticity points. The three-dimensional shapes of the eddies are now made clear: in the region containing the Gulf Stream, several strong eddies reach very deeply into the ocean, while smaller eddies remain near the surface, and the Gulf Stream itself only dominates near the surface. May 2011. [http://viz.lanl.gov/publication/movies/Eddies.m4v, http://viz.lanl.gov/publication/movies/Eddies-Globe.mp4, http://viz.lanl.gov/publication/movies/Eddies-Atlantic-Globe.mp4, http://viz.lanl.gov/publication/movies/Eddies-Orbit.mp4]



# Appendix E GUI Main Menu

The UV-CDAT main menu is divided into six sections:

1. **UV-CDAT**: Common high level items for the app, varies based on platform
    a. **About**: credits, version, and other information about uvcdat
    b. **Preferences**: spawns the preferences dialog widget
    c. **Quit**: exit UV-CDAT
2. **File**: contains common file operations, mostly related to project files
    a. **New Project**: creates a new project
    b. **Open Project**: open a saved project file
    c. **Close Project**: closes the current active project
    d. **Save Project**: saves project to user specified file
    e. **Save Project As**: same as save, but choose a new filename everytime
3. **Edit**:
    a. **CDMS Cache**: brings up the cache manager widget for CDMS files
    b. **Undo**: undo last action in currently selected cell
    c. **Redo**: redo last action in currently selected cell
4. **View**:
    a. **Projects**: toggles the visibility of the projects widget, which is visible by default
    b. **Templates**: toggles the visibility of the templates widget (default: visible)
    c. **Plots and Analysis**: toggles the visibility of the plots widget (default: visible)
    d. **Variables**: toggles the visibility of the variables widget (default: visible)
    e. **Calculator**: toggles the visibility of the calculator widget (default: visible)
    f. **Visualization Properties**: toggles the visibility of the plot properties widget (default: hidden)
5. **VisTrails**: contains actions related to the VisTrails window
    a. **Builder**: brings up the VisTrails builder window
    b. **Console**: brings up the VisTrails console widget
6. **Tools**: contains many kinds of climate related actions that are performed on the selected variables in the variables widget.
    a. **Time Tools**: contains various time related operations that are performed on the selected variables in the variables widget
        i. **Bounds Set**:
            1. **Set Bounds For Yearly Data**:
            2. **Set Bounds For Monthly Data**:
            3. **Set Bounds For Daily Data**:
            4. **Set Bounds For Twice-daily Data**:

5. **Set Bounds For 6-Hourly Data**:
6. **Set Bounds For Hourly Data**:
7. **Set Bounds For X-Daily Data**:

ii. **Extract**:

1. **Annual Means**:
2. **Seasonal Means**:
3. **DJF**:
4. **MAM**:
5. **JJA**:
6. **SON**:
7. **Monthly Means**:
8. **JAN**:
9. **FEB**:
10. **MAR**:
11. **APR**:
12. **MAY**:
13. **JUN**:
14. **JUL**:
15. **AUG**:
16. **SEP**:
17. **OCT**:
18. **NOV**:
19. **DEC**:

iii. **Climatology**:

1. **Annual Means**:
2. **Seasonal Means**:
3. **DJF**:
4. **MAM**:
5. **JJA**:
6. **SON**:
7. **Monthly Means**:
8. **JAN**:
9. **FEB**:
10. **MAR**:
11. **APR**:

12. **MAY**:
13. **JUN**:
14. **JUL**:
15. **AUG**:
16. **SEP**:
17. **OCT**:
18. **NOV**:
19. **DEC**:

iv. **Departures**:

1. **Annual Means**:
2. **Seasonal Means**:
3. **DJF**:
4. **MAM**:
5. **JJA**:
6. **SON**:
7. **Monthly Means**:
8. **JAN**:
9. **FEB**:
10. **MAR**:
11. **APR**:
12. **MAY**:
13. **JUN**:
14. **JUL**:
15. **AUG**:
16. **SEP**:
17. **OCT**:
18. **NOV**:
19. **DEC**:

b. **Regridding**:

i. **ESMF**:

1. **Linear**:
2. **Conservative**:
3. **Patch**:

ii. **LibCF**:

iii. **Regrid2**:

c. **Spatial Tools**:
    i. **Extract**:
        1. **Transcom Regions**:
        2. **Koeppen-Geiger**:
        3. **Dominant "pure" PFT**:
        4. **Vegetated Mask**:
d. **Statistics**:
    i. **Autocorrelation**:
    ii. **Autocovariance**:
    iii. **Correlation**:
    iv. **Covariance**:
    v. **Geometric Mean**:
    vi. **Lagged Correlation**:
    vii. **Lagged Covariance**:
    viii. **Linear Regression**:
    ix. **Mean**:
    x. **Mean Absolute Difference**:
    xi. **Median**:
    xii. **Rank (in %)**:
    xiii. **Root Mean Square**:
    xiv. **Standard Deviation**:
    xv. **Variance**:
e. **Vertical Dims**:
    i. **Linear interpolation**:
    ii. **Log-Linear interpolation**:
    iii. **Reconstruct Pressure: P=B*Ps+A*P0**:
f. **Filters**:
    i. **121 Filter**:
    ii. **Custom Filter**:
    iii. **Running Average**:
g. **Not Self Describing Files**:
    i. **Read ASCII File**:

**Additional Widgets**

Many of the actions in UV-CDAT require additional dialogs and widgets to get more information or allow users to change some configuration related to the operation.

**Load Variable**: This widget is spawned when the user clicks on either the add or the edit icon in the variable widget's toolbar. Its primary function is to assist in the loading and editing of data and variables. It contains five tabs as follows:

- **File**: This tab contains widgets to load variables from files, and customize the extents of the data being loaded. It contains the following elements from top to bottom:
    - **File**: This text input line specifies the file path to load. URLs can also be entered here as long as the server end point supports it. To the right of the text input is a browse button that brings up a file chooser dialog tailored to the platform UV-CDAT is running on.
    - **Variable(s)**: A dropdown widget that is populated with all of the variables available to load from the file. The desired variable must be selected from this list in order to load it.
    - **History**: A file list containing all of the recently opened files during the session.
    - **Bookmarks**: Another file list that persists between sessions. Users can drag frequently opened files from the history list to the bookmarks list in order to easily reuse the file in future UV-CDAT sessions.
    - **Dimensions**: In the bottom half of the file tab are the dimensions widgets, one for each axis in the selected variable, that allow users to further refine the extents of the data they wish to load. Each dimension widget consists of the following items from left to right:
        - **Axis**: this drop down serves several purposes:
            1. Select extents by value (default) or by index in the data array
            2. Load axis values as its own variable
            3. Load the weights of the axis values as its own variable
            4. Replace the axis values
            5. Reorder (or remap) the Dimensions widgets e.g. (change axis Z from time to longitude)
        - **Values**: This dropdown menu can be used to select upper and lower bounds for the axis. Only data between these bounds (inclusive) will be loaded. It is linked to the sliders widget.
        - **Sliders**: These slider widgets offer an additional way to set the extents of an axis
        - **Pin**: Clicking this button saves the extents so the same can be used to load multiple variables from multiple files. Otherwise as soon as a new variable is selected, these are reset to their default postions.
        - **Aggregator**: This dropdown provides options to apply statistical operations over an axis. These include:
            1. **Def**: By default, no operations are used
            2. **Sum**: Summation
            3. **Avg**: Average
            4. **Wgt**: Weighted Average
            5. **Gtm**: Geometrical Mean
            6. **Std**: Standard Deviation

    - **Select Region Of Interest (ROI)**: This button brings up the ROI dialog to visually select the data extents on a map. See the Region of Interest section of 4.1.6.8.

- **ESGF**: This tab allows users to access data directly from ESGF. It is designed to be similar in look and functionality to the ESGF online portal.
- **OpenDAP**: This tab contains a tree like browser to load data from remove data servers. By default, contains the DODS Catalog, but others can be added and removed using the two buttons on the bottom left. Clicking on a data server fetches its catalog, which are either directories or files. After the data is fetched, the item can be expanded to view its contents. The directories that it fetches work the same way. Clicking on a data file displays its information. With a file selected, clicking Open copies the URL to the File line edit in the File tab, and brings that tab to the fore front to finish the data loading process like normal.
- **Edit**: This tab is disabled during data loading. To access it, users must select the edit icon in the variables widget after a variable has been selected. It contains the following sections from top to bottom:
    - **Edit Projects**: Using these checkboxes, you can change which projects the variable will be avaialbe to
    - **Variable Attributes**: Modify, create, and delete attributes of the variable.
    - **Axes Attributes**: Modify, create, and delete attributes on specific axes.
    - **Dimensions**: These are the same as those in the file tab. See File tab description above.
- **Info**: Displays lots of detailed information about a variable.

**Region of Interest**: This widget displays a map, either with or without gridlines, and allows users to draw a rectangle around a region of interest. It is directly tied to the axes widgets in load variable widget, which are updated to match the selection made with this widget. The “Select Region of Interest (ROI)” button at the bottom left of the load variable widget is used to spawn this widget.

**Plot Properties**: This widget can be viewed by cliking the pencil icon in the spreadsheet toolbar sith a single cell selected that contains a visualization. It contains elements related to the plot, and allows user to change parameters and plot configuration. Each package has unique settings and configuration, so this widget will change depending on the plot in the selected spreadsheet cell.

**Variable Plot Queue Manager**: clicking the list icon in the spreadsheet toolbar spawns this. It contains two lists, one for the variable queue and the other for the plot queue. Selecting them and clicking the remove button can then remove queued items.

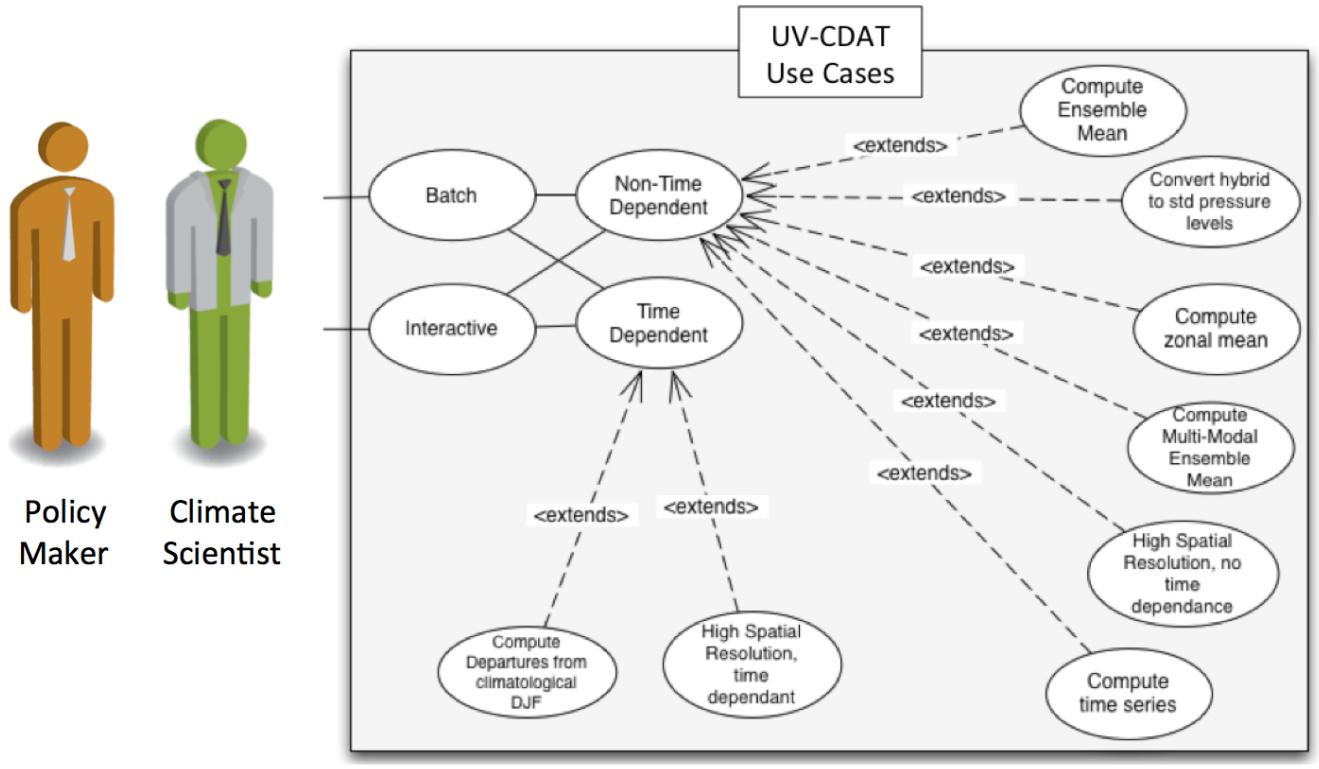
**PCMDITools**: Some of the operations in the PCDMITools menu spawn an intermediate widget to complete the operation. The widget generally displays some documentation on how the operation works, and provides a few settings related to the operation for users to change as needed.

**Animation**: The widget is displayed by clicking the animation icon in the spreadsheet toolbar. By default it will be docked in the bottom left in a new tab along with the plots and templates widgets. Clicking on the play button will initialize the animation, which may take some time depending on how much data needs to be pre-rendered. Once all of the frames have been rendered, they can be played back at multiple speeds.

**CDMS Cache Manager**: This is spawned by selecting Edit > CDMS Cache from the main menu bar. It shows a list of cached data files. Items can be removed from the cache by selecting them and clicking “delete” in the bottom left.

# Appendix F Use Cases

In general, the use-case diagram for UV-CDAT, shown below, includes an actor (climate scientist or policy-maker) and specific use cases (oval shapes). For high-level UV-CDAT system functions and requirements, actors are entities that interact with the system, while use cases are the system functions the actors involve in their research. The use-case diagram tells "WHAT" the system should do, but does not illustrate "HOW TO" complete the action.




*Use case diagram for UV-CDAT.*

Twelve different use cases have been defined.

### *Use Case 1: High spatial resolution, parallel, image sequence production*

**Actor:** Climate/Data Scientist
**Use Scenario** The user chooses to produce an image sequence by producing one picture per time step.

- More than one processor is used to iterate through N number of 3-D spatial fields saved at time steps, 0,1…,N-1, run a filter on each, and render N sequential 3D images.
- The data should be high resolution, i.e., 1/10-degree global ocean (3600x2400x42).
- The data should be structured, i.e., rectilinear grid.
- The data for each time step should be divided spatially and distributed across more than one processor.
- The data should have a filter, run in parallel, which produces geometry, which is then rendered and composited to produce a single image.

**Alternative Paths:**
**Exceptional Cases:**
**Frequency:** High
**Criticality:** High
**Risk:** High

### *Use Case 2: High spatial resolution, parallel, time average*

**Actor:** Climate/Data Scientist
**Use Scenario:** A user requests a data set be averaged across a number of time steps.

- More than one processor is used to iterate through N number of 3-D spatial , run a filter on each, and render a single 3D image.
- The data should be high resolution, i.e., 1/10-degree global ocean (3600x2400x42).
- The data should be structured, i.e., rectilinear grid.
- The data for each time step should be divided spatially and distributed across more than one processor.
- The operation should return a single time step average of the same spatial resolution as the time series that contains the positional temporal average.
    - For example 3 time steps = {1,2,3}, {1,2,6}, {7,8,9}
    - Return value is {3, 4, 6}

**Alternative Paths:**
**Exceptional Cases:**
**Frequency:** Medium
**Criticality:** High
**Risk:** High

### *Use Case 3: Compute ensemble mean*

**Actor:** Climate/Data Scientist
**Background:** Usually for each experiment, the modeling groups will produce multiple instances (or realizations) of the same experiment to obtain an ensemble that encompasses a range of chaotic behavior of the system. Statistics computed from the ensemble, such as the average and moments, are often used to understand natural system variability and the system response to changes in forcing. Typically for AR4 there were at most 10 such realizations per experiment, there will likely be more in the future.
**Use Scenario:**

- Step 1: identify all runs for the ensemble and the files for each run.
    - An example AR4 ncar-ccsm3 model in the box below. The one CDAT would actually use is highlighted; these xml files are recognized by CDAT and have aggregation of all the .nc files. So in this case we would need to open these 9 files.
- Step 2: identify common time between each runs
    - In this case all instances cover the same 1870 thru 1999 period. But we cannot assume this. In the case they do not cover the same periods, a decision needs to be made: Compute over the common period only, or compute over the "union" of times and weight each time according to the number of models.
- Step 3: average ensemble members over common time
    - In this case basically for each time step add all 9 instances together. WARNING: We have to be careful about missing values, for 3D fields some "bellow ground" values might be masked. These will be different for each time step and each model. That means we cannot assume 9 at the end will divide the sum! (Note: It may be true across models, that vertical grids are the same for given models in a scenario. However, if they are not, then we will need to interpolate to a consistent vertical grid.)
- Step 4: dump out result: array or to file (in case memory is not big enough)

**Difficulties:**

- Identify files to use
- Identify which time period each instance covers and decide what to do if not common across members
- Pay attention to missing values

- **Alternative Paths:**
- Compute percentiles (median or 10 percentile for example) instead of mean.
- **Exceptional Cases**
- If the xml are not present then "generate them" (makes more sense) or "do without them"

**Frequency:** Medium
**Criticality:** Medium
**Risk:** Medium

### *Use Case 4: Compute average multi-model ensemble mean*

**Actor:** Climate/Data Scientist
**Background:** This is basically the same problem as case 3, but each instance is on a different grid. We average over all instances, each of which is the output of a different model, to compute the "mean model" result, which as been shown to be superior to all models when compared to observations.
**Use Scenario:** User has coinciding time steps from a variety of models, run with the same starting conditions. For each time step, an average of the ensemble of data sets should be calculated, which requires regridding and/or registration of the data sets.

- Step 1: identify all models for this ensemble
- Step 2: identify common time between each runs
- Step 3: identify a target grid to which all model will be regridded
- Step 4: average ensemble members over common time and on common grid
- Step 5: dump out result

**Difficulties:**

- Identify files to use.
- Identify which time period each member covers and decide what to do if not common across members.
- Pay attention to missing values.
- Decide on a regridding method and a target grid that make sense.

**Alternative Paths:**

- Calculate the median (usually preferred to the mean model) (eliminates outliers).
- Calculate the standard deviation or other statistics

**Exceptional Cases:**

- If the XML are not present then "generate them" (makes more sense) or "do w/o them".

**Frequency:** Medium
**Criticality:** Medium
**Risk:** Medium

### *Use Case 5: Compute departures from climatological boreal winter (DJF)*

**Actor:** Climate/Data Scientist
**Background:** Scientists frequently focus on departures (or anomalies) from seasonal or monthly climatology. The first step is to produce a model's climatology of monthly and seasonal averages through an annual cycle.
**Use Scenario:**

- Step 1: identify file(s) covering the time period for the climatology.
- Step 2: compute climatology over period of reference (climatology is defined usually as a 30 years average)
    - Average the December (D), January (J), and February (F) files over the period of reference
    - Note that boreal winter spans calendar years. For example the first DJF only has 2 months, the last one only has 1 month. Rules need to be put in place to decide how to

weight these cases! WARNIG 2: Time bounds MATTER, for data set expressed in "months since" Jan and February have the same length (1 month), but for data set expressed in "days since" Jan has a length of 31 whereas Feb has a length of 28 (and even 29 every 4th years) once each DJF has been computed, average them all together, that is our climatological DJF

- Step 3: identify file(s) covering departures to be computed
  - Can be longer than ref period
- Step 4: compute seasonal means
  - Repeat the first part of the process for computing the climatology, obtaining the "Seasons"
- Step 5: remove climatology from step 4
- Step 6: dump out result

**Difficulties:**

- Time bounds matter!
- Missing values!

**Alternative Paths:**

- Annual cycle or other seasonal period, instead of DJF

**Exceptional Cases:**
**Frequency:** Medium
**Criticality:** Medium
**Risk:** Medium

### *Use Case 6: Convert from hybrid to standard pressure levels*

**Actor:** Climate/Data Scientist
**Background:** The stand vertical dimension of 3D spatial meteorological data is hydrostatic pressure with units of hPa. Many atmospheric models use a different vertical coordinate and their output is not stored on pressure levels (there are 17 standard pressure level in AR4). Consequently, model results are usually interpolated back to standard pressure levels. As an example, the conversion from hybrid to standard pressure levels is presented.
**Difficulties:**

- The horizontal variation of hydrostatic pressure is not orthogonal to the surface, resulting in constant pressure surfaces that intersect the lower boundary.

**Alternative Paths:**

- Different vertical coordinate systems

**Exceptional Cases:**

- Step 1: Identify file(s) covering period to convert
- Step 2: compute "pressure" field over time covered
  - Conversion requires the hybrid coefficient vectors A and B, from the model and the global surface pressure field (2d lat/lon): compute P = B*Ps + A*P0
- Step 3: For each output level to a linear/log interpolation from pressure field to actual output pressure (mask out below ground)
- Step 4: Dump out result

**Frequency:** Medium
**Criticality:** Medium
**Risk:** Medium

### *Use Case 7: Compute a time series of a regional average*

**Actor:** Climate/Data Scientist

**Details:** Read in data over a spatial region and return an average value at a given time. This is typically a drop in dimensionality
**Use Scenario:**

- Step 1: Identify needed file(s) to cover time period
- Step 2: Identify non-time dimensions
- Step 3: Average over non-time dimensions for each time value in the series, Note: order of calculation may be important because of treatment of missing values
- Step 4: Create time-series file of areal averages

**Difficulties:**

- Handling of missing values and area weights for grid points in averaging areas

**Alternative Path:**

- Single location, difference between 2 locations (North Atlantic Oscillation, etc.)

**Exceptional Cases:**
**Frequency:** Medium
**Criticality:** Medium
**Risk:** Medium

### *Use Case 8: Computing a zonal mean*

**Actor:** Climate/Data Scientist
**Background:** many fields have a strong latitudinal dependence, which can be easier to discern once they have been zonally averaged.
**Use Scenario:**

- Step 1: identify file(s) to cover time period
- Step 2: identify "zonal" dimension (longitude) and bounds for each zonal band
- Step 3: average over "zonal" dim
- Step 4: dump out result

**Difficulties:**

- Non latitude/longitude grids will require either interpolation or alternative averaging

**Possible variation:**

- Use level instead of longitude average

**Frequency:** Medium
**Criticality:** Medium
**Risk:** Medium

### *Use Case 9: Batch Processing*

**Actor:** Climate/Data Scientist
**Use Scenario:** A program is written that uses the UV-CDAT framework to produce a product. This program is submitted to a job scheduler on a supercomputer and writes it's product to a file system.
**Alternative Paths:**
**Exceptional Cases:**
**Frequency:** High
**Criticality:** High
**Risk:** High

### *Use Case 10: Interactive Processing*

**Actor:** Climate/Data Scientist
**Use Scenario:** A Graphical User Interface that uses the UV-CDAT framework to load and transform data sets is used to interactively explore one or more data sets using distributed memory back-end processing.

**Alternative Paths:** GUI runs in serial.
**Exceptional Cases:**
**Frequency:** High
**Criticality:** High
**Risk:** High

### *Use Case 11: Time Dependent Processing*

**Actor:** Climate/Data Scientist
**Use Scenario:** A processing step is called that requires an entire time series to be processed in order to produce a result.
**Alternative Paths:**
**Exceptional Cases:**
**Frequency:** High
**Criticality:** High
**Risk:** High

### *Use Case 12: Time Independent Processing*

**Actor:** Climate/Data Scientist
**Use Scenario:** A processing step is called that only requires a single time step to be read to produce a result.
**Alternative Paths:**
**Exceptional Cases:**
**Frequency:** High
**Criticality:** High
**Risk:** High

# Appendix G Acronyms

| | |
|---|---|
| AMWG | Atmospheric Modeling Working Group |
| API | application programming interface |
| BER | Department of Energy's Office of Biological and Environmental Research |
| CAM | Community Atmosphere Model |
| CF | Climate and Forecast |
| CDAT | Climate data analysis tools |
| CDMS | Climate Data Management System |
| CESM | Community Earth System Model |
| CF | configuration files |
| CLM | Community Land Model |
| CMIP5 | Climate Model Intercomparison Project, Phase 5 |
| CMOR | Climate Model Output Rewriter |
| DOE | Department of Energy |
| DV3D | Data Visualization 3D |
| EC | Executive Committee |
| ECMWF | European Center for Medium-Range Weather Forecasts |
| EDEN | Exploratory Data analysis Environment |
| ESM | Earth System Model |
| ESMF | Earth System Modeling Framework |
| ESGF | Earth System Grid Federation |
| FAQs | Frequently Asked Questions |
| GEOS5 | Goddard Earth Observatory System Model, Version 5 |
| GRIB | GRIdded Binary |
| GSFC | Goddard Space Flight Center |
| GUI | graphical user interface |
| HDF | hierarchical data format |
| I/O | input/output |
| IPCC | Intergovernmental Panel on Climate Change |
| IS-ENES 2 | Infrastructure for the European Network for Earth System Modelling |
| LANL | Los Alamos National Laboratory |
| LBNL | Lawrence Berkeley National Laboratory |

| | |
|---|---|
| LEI | Leaf Area Index |
| LLNL | Lawrence Livermore National Laboratory |
| LMWG | Land Model Working Group |
| MMF | Multiscale Modeling Framework |
| MMM | multi-model mean |
| MHC | Meridional Heat Transport |
| MOC | Meridional Overturning Circulation |
| MPAS | Model for Prediction Across Scales |
| NASA | National Aeronautics and Space Administration |
| netCDF | network Common Data Format |
| NCCS | NASA Center for Climate Simulation |
| NCL | NCAR Command Language |
| NCO | NetCDF Operators |
| NOAA | National Oceanic and Atmospheric Administration |
| NYU-Poly | Polytechnic Institute of New York University |
| OMWG | Ocean Model Working Group |
| OPeNDAP | Open-source Project for Network Data Access Protocol |
| OPM | Open Provenance Model |
| ORNL | Oak Ridge National Laboratory |
| PCWG | Polar Climate Working Group |
| PP | post processing |
| POP | Parallel Ocean Program |
| PROV | Provenance |
| REST | Representational State Transfer |
| RMSE | root mean-square error |
| SC | Steering Committee |
| TC | Technical Committee |
| UCAR | University Corporation for Atmospheric Research |
| UQ | uncertainty quantification |
| UQTk | UQ Toolkit |
| URB | UV-CDAT Review Board |
| UV-CDAT | Ultra-scale Visualization Climate Data Analysis Tools |
| VCS | Visualization Control System |

| | |
|---|---|
| ViSUS | Visualization Streams for Ultimate Scalability |
| VTK | Visualization Toolkit |
| WCRP | World Climate Research Programme |
| WGCM | Working Group on Coupled Models |
| WGNE | Working Group on Numerical Experimentation |
| WRF | Weather Research and Forecasting model |
| W3C | World Wide Web Consortium |