Ultrascale Visualization of Climate Data

Ultrascale Visualization Climate Data Analysis Tools Project Team

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

Fueled by exponential increases in the computational and storage capabilities of high-performance computing platforms, climate simulations are evolving toward higher numerical fidelity, complexity, volume, and dimensionality. These technological breakthroughs are coming at a time of exponential growth in climate data, with estimates of hundreds of exabytes by 2020.1

To meet the challenges and exploit the opportunities that such explosive growth affords, a consortium of four national laboratories, two universities, a government agency, and two private companies formed to explore the next wave in climate science. Working in close collaboration with domain experts, the Ultrascale Visualization Climate Data Analysis Tools (UV-CDAT) project aims to provide high-level solutions to a variety of climate data analysis and visualization problems:

- **Dealing with big data analytics.** Climate science is no different from other domains in its pursuit of solutions to process, analyze, and visualize massive datasets.
- **Sensitivity analysis.** The community must be able to push ensemble analysis, uncertainty quantification, and metrics computation to new boundaries.
- **Heterogeneous data sources.** Climate science data comes from simulations, observations, and reanalysis. Any visualization and analysis solution must unify these sources.
- **Reproducibility.** All science must support systematic data maintenance by providing provenance to ensure reliable and persistent links between workflows.
- **Multiple disciplinary domains.** Complexity stems from the need to incorporate a broad nexus of climate and other related science domains such as climate adaptation and mitigation for water, energy, and agriculture conservation.
- **Flexible, scalable architecture.** Any unifying structure must be able to incorporate both existing and future software components with minimal or no infrastructure modification.

As the “UV-CDAT Consortium” sidebar describes, the project integrates computational and domain scientists from all sectors in tackling these problems, with the ultimate goal of building an ultrascale data analysis and visualization system that will empower novel data exchanges. The project, which began in late 2010 and is expected to complete in late 2013, has met several major objectives. As of August 2013, the team has

- released the official UV-CDAT system version 1.4 (http://uv-cdat.org);
- addressed projected scientific needs for data analysis and visualization;
• extended UV-CDAT to support the latest regridding by interfacing to the Earth System Modeling Framework (ESMF; www.earthsystemmodeling.org) and Climate and Forecast library (LibCF; www.unidata.ucar.edu/software/libcf); and
• successfully supported ongoing climate model evaluation activities for US Department of Energy (DOE)’s climate applications and projects (www.climatemodeling.science.energy.gov/projects), such as the Intergovernmental Panel on Climate Change (IPCC) assessment report and Climate Science for a Sustainable Energy Future (CSSEF).

UV-CDAT brings to bear many capabilities that directly address climate scientists’ needs, including parallel streaming statistics, optimized parallel I/O, remote interactive execution, workflow capabilities, and the automatic capture of data provenance. Already, the UV-CDAT architecture is enabling new capabilities in visualization, regridding, and statistical analysis.

ARCHITECTURAL OVERVIEW
One of our main project objectives is to assist in ultrascale scientific analysis and visualization. In light of this goal, one of UV-CDAT’s strongest features is its Python-based framework, which integrates disparate technologies under one overarching infrastructure. Figure 1 shows the framework’s conceptual design.

Using standard common protocols and application programming interfaces (APIs), the framework allows UV-CDAT to integrate 70-plus (to date) software components. The primary software stack comprises climate data analysis tools (CDAT), VisTrails, Data Visualization 3D (DV3D), and ParaView. DV3D and ParaView are two of five analysis and visualization tools that have gained considerable popularity. The other three are Visit (https://wci.llnl.gov/codes/visit/home.html), Visualization Streams for Ultimate Scalability (ViSUS; http://visus.us), and Visualization Toolkit (VTK; www.vtk.org).

However, none of these tools are customized for climate research, forcing climate scientists to build individual analyses around each tool, and in the process often recreating code developed by other scientists. CDAT provides a more direct approach because it integrates these tools into the UV-CDAT framework, which essentially customizes them to process climate data and common analyses.

Scientists must also be able to create integrated analysis and visualization pipelines that interweave multiple tools, libraries, and services. To aid that creation, UV-CDAT relies on the capabilities provided by VisTrails, Provenance storage, and Workflow.

Figure 1. The Ultrascale Visualization Climate Data Analysis Tools (UV-CDAT) project framework. By uniting common protocols and APIs in either a loose coupling or tight integration, the framework enables scientists to understand climate and observation data across models, providing the breadth of understanding needed to interpret climate change.
on VisTrails, an open source provenance and workflow system.

Using an open source, object-oriented Python scripting language that is easy to learn, CDAT links disparate software subsystems and packages to form an integrated environment for data analysis. The integration concept is simple and flexible, allowing researchers to interchange the application’s parts and expand for the future. Through Python, CDAT provides a full-featured scripting language with a variety of user interfaces, including command-line interactions, standalone scripts (applications), and a GUI.

The CDAT core consists of the Climate Data Management System (CDMS), large-array numerical operations, climate analysis and diagnosis, and statistics. These subsystems provide climate scientists with on-demand analysis and diagnostics, which facilitate a better understanding of climate processes and effects.

By combining these CDAT core subsystems with enhanced data visualization engines, a provenance workflow system, and an integrated framework, UV-CDAT can offer end-to-end solutions for data management, analysis, and visualization for the ultrascale datasets.

The UV-CDAT framework couples software infrastructures either tightly for greater system functionality and communication with other components or loosely for fast integration. Tightly coupling CDAT’s core subsystems with the VTK/ParaView infrastructure, for example, enables seamless high-performance parallel streaming data analysis and visualization of massive climate datasets. Loosely coupled integration then allows the addition of other tools such as Matlab and customized data analysis tools with little or no modification. Within both paradigms, UV-CDAT provides data provenance, workflow capture, and mechanisms to support data analysis via the VisTrails infrastructure.

In addition, users can add custom functionality through an intuitive interface that includes tools for workflow analysis and visualization construction. To augment these capabilities, we include links to the R statistical analysis environment (EDEN), and VisIt. All these functions and tools are integrated under a Python- or Qt-based architecture (http://qt-project.org).

**DATA, METADATA, AND GRIDS**

Data formats play an integral role in consolidating geoscience information. In climate research, data comes from model simulations, instruments, or observations as well as metadata, such as how the data was generated, what it represents, and what is to be done with it. Most collections of model runs, observations, and analysis files provide a uniform data access interface to conventional formats, such as the Network Common Data Form (netCDF), Hierarchical Data Format (HDF), Gridded Information in Binary form (GRIB), and Post-Processing (PP) format. For climate modeling simulation and observation, groups are increasingly storing their data in netCDF, whereas the climate and forecast (CF) methodology is becoming the de facto convention for defining metadata. Combining netCDF and CF into one convention, netCDF-CF, makes it easy for scientists to compare and display a range of geoscience datasets and to decide which quantities are comparable. This capability supports building applications with powerful extraction, regridding, and display functionalities. With UV-CDAT’s Climate Model Output Rewriter (CMOR), scientists can easily obtain properly formatted data, which will encourage the smooth adoption of the netCDF-CF convention.

**ANALYSIS**

Analysis consists of regridding (interpolating data from one grid to another), exploring the data and generating hypotheses, parallel processing, and capturing provenance information.

**Regridding**

Our recent user survey revealed that regridding is among the most widely used features in climate data analysis tools. In UV-CDAT, we extended this feature to support curvilinear grids, which ocean and atmospheric models often rely on to overcome numerical stability issues at the North and South Poles. Examples of curvilinear grids are the displaced or rotated pole grid and the tripolar grid, which some ocean models use to remove the North Pole singularity from the grid. The block-structured, cubed-sphere grid that some atmospheric models rely on is another example of a curvilinear grid with no singularity at the poles.

Regridding Earth data is a conservative interpolation that presents unique challenges. The data might have missing or invalid values, such as ocean data values that fall on land. Also, although conservative interpolation is the method of choice for cell-centered data, it can be significantly more numerically intensive than nodal interpolation.
We have addressed these challenges by leveraging multiple existing interpolation libraries, and by designing a single Python regridding interface that supports multiple interpolation tools, such as ESMF and the Spherical Coordinate Remapping and Interpolation Package (SCRIP; http://climate.lanl.gov/Software/SCRIP), and methods, which at present include linear nodal, quadratic nodal, and conservative. On the basis of grid type (rectilinear or curvilinear) and data type (nodal or cell), the interface automatically selects the most appropriate tool and method for the task.

**Exploratory data analysis and hypothesis generation**

Another important aspect of the UV-CDAT framework is its ability to provide users with the means to quickly explore massive amounts of data to form new hypotheses and verify simulation data. Through one interface, scientists can now access CDAT, ParaView, VisIt, and DV3D—four important toolkits for visual data exploration, as Figure 2 shows. UV-CDAT thus provides all traditional visualization methods, such as slicing, volume rendering, and isosurfac- ing, as well as the ability to explore long time series and create animation sequences. Through a spreadsheet paradigm, scientists can combine a variety of 2D and 3D plots.

**Parallel processing**

As resolution increases, climate models are continuously improving numerical fidelity—resolution and number of variables within a smaller region—to the point that the memory footprint becomes too large for the data to reside on a single processor. Most platforms can accommodate the loading and processing of only one 3D variable at a 10-km resolution in a single time step. To help users handle such memory-heavy processing and other numerically intensive operations, we extended the CDMS arrays in CDAT to allow remote memory access within the scripting UV-CDAT environment. The new functionality supports remote data access through a get method, which takes the remote processing rank and a tuple that uniquely represents a slice of the data to be fetched.

We used the mpi4py module (http://mpi4py.scipy.org) to implement these functions in Python, relying on recent one-sided communication enhancements to the MPI-2 standard. The resulting implementation of distributed array functionality works in any number of dimensions and for multiple data types. Although simple, an implementation based on remote memory access is more flexible than one based on point-to-point send or receive calls because the process that exports data does not need to know which process to send data to, and each process can access data residing on any other processor.

**Capturing provenance information**

During analysis, VisTrails automatically captures provenance information, making it possible to reproduce and share results, and thereby reducing the effort to manage scripts and data files. Each analysis has a corresponding workflow that is updated when the analysis changes, such as when the user changes a parameter or introduces an intermediate step. Because updates are incremental, VisTrails keeps all versions of the analysis as provenance information. Figure 3a is an example of an automatically
generated workflow for regridding a variable and plotting it using the boxfill plot type from the CDAT library. Figure 3b shows the resulting plot.

**WORKFLOW GENERATION**

In the UV-CDAT framework, workflow is captured and stored through community tools, such as software compilation controllers, dashboards, and testing tools, and analyzed through open source systems that support data exploration and visualization.

**Community tools and environments**

Developing large-scale software requires a rigorous process that takes a concept through design, development, implementation, code integration, testing, and user delivery in a robust and systematic way. This process requires monitoring to ensure quality and involves tools such as the open source CMake, CTest, and CDash packages. In the UV-CDAT framework, community tools include these packages as well as Github, a repository for data, documentation, and code.

- CMake uses simple platform- and compiler-independent configuration files to control software compilation. A cross-platform build system based on CMake generates native makefiles and workspaces.
- CTest, a testing tool distributed as a part of CMake, automates code update, configuration, build, and test operations.
- CDash, an open source, Web-based software-testing server, aggregates, analyzes, and displays the results of software testing processes that clients submit.

UV-CDAT’s software process supports agile development methods and is motivated by test-driven development approaches. A similar process in use by thousands of software systems has scaled to tens of millions of lines of code.

**Community visualization and analysis**

Visualization and analysis requires systems that support data exploration and visualization, interactive visualization, rapid access to large datasets, postprocessing, and statistical computing, as well as a flexible, encompassing interface.

**VisTrails.** VisTrails lets users specify the computational processes that will integrate existing applications, loosely coupled resources, and libraries simply by clicking buttons or dragging variables and plot types. VisTrails’ API converts these events into workflow operations, such as module creation and parameter changes, and captures them as provenance information. It then notifies the system to update the plots and the GUI, as necessary. Scientists can also use the workflow builder to edit workflows directly and to create new plots.

VisTrails captures and maintains a detailed history of the steps followed and data derived during an exploratory task, maintaining the provenance of the workflows that derive these data products as well as their executions. Using a thin Python interface encapsulated by a set of VisTrails modules, developers can expose their libraries (in any language) to UV-CDAT, making it simple to integrate tools and libraries as well as to quickly prototype new functions.

**DV3D.** DV3D, a VisTrails package of high-level modules for UV-CDAT, provides user-friendly workflow interfaces for advanced climate data visualization and analysis. DV3D provides the interfaces, tools, and application integration required to make VTK’s analysis and visualization power readily accessible to scientists without exposing details such as actors, cameras, and renderers. It can run as a
As part of our work on the UV-CDAT framework, we added two climate-specific operations to VisIt: peaks-over-threshold computing and extreme value analysis.
threshold computing and extreme value analysis. Both these operations use GNU-R scripts at their core and interface with VisIt and UV-CDAT through the VTK-R bridge.

Figure 4 shows a computation of the extreme value analysis operation using VisIt-R and temperature data rendering using VisIt, as well as DV3D, CDAT, and ParaView plots. Scientists use the operation and model output and observations to estimate past and future changes in extreme precipitation and other climate variables.

**Statistical computing.** R is a widely used statistical computing package, so incorporating it into VisIt and UV-CDAT provides access to the many statistical analysis algorithms that are at the core of climate analysis work. VisIt uses custom R scripts to compute several climate-related operations, and we are actively working on expanding R procedures in UV-CDAT.

**EDEN.** Developed collaboratively with climate researchers on the CSSEF project, EDEN blends interactive information visualization techniques with automated statistical analytics to effectively guide the scientist to the most significant relationships.

**Workflow examples**

Average, a simple model-run diagnostic procedure to compute the average value of a climate variable (such as

---

**Figure 4.** Extreme value analysis plotting using an array of visualization tools, such as DV3D (upper left), VisIt-R plot (lower left), CDAT plots (middle), VisIt-R (upper right), and ParaView (lower right). Using intuitive drag-and-drop operations, scientists can create, modify, copy, rearrange, and compare visualizations.
temperature) and hashvar, a typical IPCC-related analysis, illustrate the overall UV-CDAT workflow—purpose, data and metadata, tools, and visual results.

**Average.** The average operation is useful in determining the average value of a variable in the input files over many months or many years, and seeing how the average varies by location. A possible application is to calculate the climatological annual cycle of rainfall or surface temperature.

From a list of netCDF files and a list of variables of interest, the operation computes the average value for each latitude and longitude point for each variable over all the input files. It then creates a new netCDF file that has the average value for each variable.

In the output file, variable \( X \) at coordinate (0,0) is the average value for all \( X \)s over the input files at coordinate (0,0).

**Hashvar.** As its name implies, hashvar is a frequency hashing operation, which is useful in spotting monthly or yearly consistent trends in the input files. For example, suppose a scientist wants to analyze if a temperature trend is present from a list of netCDF files. Hashvar determines the minimum and maximum values at each latitude and longitude point across all the files, with the bucket sizes based on the minimum, maximum, and selected number of bins, as well as how often a given latitude and longitude point is within a bucket range for all the files.

Hashvar also creates new netCDF files; the number of files depends on the number of allowed variables. The new files have \( X \) new variables per variable of interest (where \( X \) is the number of bins) that show the frequency for each latitude and longitude point over the set of input files.

In the output files, variable \( \text{var}_3 \) at coordinate (0,0) is the number of occurrences of \{bin size 2\} through \{bin size 3\} of variable \( \text{var} \) at coordinate (0,0) in all the input files.
alize time and space. This unique view enables researchers to see model error growth, and allows first glimpses of model error attribution.

As geoscience datasets continue to expand in size and scope, there is a greater need to perform data analysis where the data is located (server-side analysis). UV-CDAT is therefore undergoing modifications to allow access to the DOE-sponsored Earth System Grid Federation infrastructure (www.esgf.org). With this modification, users can access petabyte archives and perform analysis and data reduction before moving the data to their site. Most important, the necessary remote operations will be routinely performed, thus freeing UV-CDAT users to concentrate on scientific diagnosis rather than on the mundane chores of data movement and manipulation.

Acknowledgments

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.

References


The UV-CDAT Project Team

Lawrence Livermore National Laboratory: Dean N. Williams, UV-CDAT principal investigator; Williams13@llnl.gov; Timo Bremer, bremer5@llnl.gov; Charles Doutriaux, doutriaux1@llnl.gov.

Los Alamos National Laboratory: John Patchett, patchett@lanl.gov; Sean Williams, seanw@lanl.gov.

Oak Ridge National Laboratory: Galen Shipman, gshipman@ornl.gov; Ross Miller, rgmiller@ornl.gov; David R. Pugmire, pugmire@ornl.gov; Brian Smith, smithbe@ornl.gov; Chad Steed, steedca@ornl.gov.

Lawrence Berkeley National Laboratory: E. Wes Bethel, ewbethel@lbl.gov; Hank Childs, hchilds@lbl.gov; Harinarayan Krishnan, hkrishnan@lbl.gov; Prabhat, prabhat@lbl.gov; Michael Wehner, MFWehner@lbl.gov.

Polytechnic Institute of New York University: Claudio T. Silva, csilva@nyu.edu; Emanuele Santos, emanuele@ia.ufc.br; David Koop, dkoop@poly.edu; Tommy Ellqvist, tommy.ellqvist@yahoo.se; Jorge Poco, jpocom@nyu.edu.

Kitware: Berk Geveci, berk.geveci@kitware.com; Aashish Chaudhary, aashish.chaudhary@kitware.com; Andy Bauer, andy.bauer@kitware.com.

Tech-X: Alexander Pletzer, pletzer@txcorp.com; Dave Kindig, kindig@txcorp.com.

NASA Goddard Space Flight Center: Gerald L. Potter, gerald.potter@nasa.gov; Thomas P. Maxwell, thomas.maxwell@nasa.gov.