Kitware

28 Corporate Drive
Clifton Park, NY 12065
518.371.3971

Accelerated Climate Modeling for Energy (ACME)

Final Scientific/Technical Report

Prepared for: The U.S. Department of Energy, Office of Science

Grant Number: DE-SC0012356

Period of Performance: August 15, 2014 – August 14, 2017

Kitware, Inc.
28 Corporate Drive
Clifton Park, NY 12065
DUNS: 01-092-6207

Aashish Chaudhary, Principal Investigator
(518) 371-3971 Ext. 506
Aashish.chaudhary@kitware.com

This material is based upon work supported by the U.S. Department of Energy, Office of Science under Award Number DE-SC0012356

Major Goals

Seven Department of Energy (DOE) national laboratories, Universities, and Kitware, undertook a coordinated effort to build an Earth system modeling capability tailored to meet the climate change research strategic objectives of the DOE Office of Science, as well as the broader climate change application needs of other DOE programs. The model development requirements are derived from the following four high-level science drivers:

1. How do the hydrological cycle and water resources interact with the climate system on local to global scales?
2. How do biogeochemical cycles interact with global climate change?
3. How do rapid changes in cryospheric systems interact with the climate system?
4. How do short-term variations in natural and anthropogenic radiatively active atmospheric constituents interact with natural variability and contribute to regional and global environmental change? The Accelerated Climate Modeling for Energy (ACME) project will build and test a next-generation earth modeling system that can be run on current and future generations of computing systems at Office of Science computing facilities, including envisioned exascale systems. The initial three-year project plan calls for the delivery to DOE and the broader research community of an open-source modeling system and the results from control and prediction simulations that document its scientific and computational performance.

The Accelerated Climate Modeling for Energy (ACME) project will build and test a next-generation earth modeling system that can be run on current and future generations of computing systems at Office of Science computing facilities, including envisioned exascale systems. The initial three-year project plan calls for the delivery to DOE and the broader research community of an open-source modeling system and the results from control and prediction simulations that document its scientific and computational performance.

Kitware contributed to the development and maintenance of two products for climate scientists: 1) vCDAT, the web interface for climate data visualization and analysis build on top of UV-CDAT, and 2) ParaView - in-situ capable large data parallel visualization tool. Additionally, we fixed bugs and improved various features as reported by the ACME team within UV-CDAT. For ParaView we have added specific capabilities needed by LANL, Sandia, and PNNL for I/O and visualization related to CAM and MPAS.

vCDAT is a desktop and web based application designed to expose all UV-CDAT functionality through an easy to use user interface. We developed the first version of this application. We designed VCS.js, an API to separating the core functionality of vCDAT (producing a visualization) from the user interface. VCS.js can produce the visualization either by calling VCS or by using client based (JavaScript) visualization libraries. We solved key performance issues affecting vCDAT. As part of the vCDAT development we fixed numerous bugs affecting the correctness and performance of the UV-CDAT. Climate Data Analysis tool (UV-CDAT) is a Python visualization and analysis library designed to meet the needs of climate scientists. It

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356

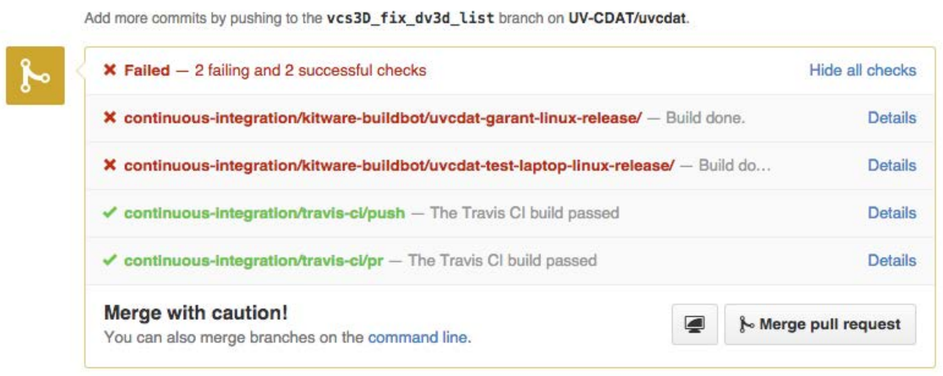
offers 1d plots such as line plots and scatter plots; plots for scalar data such as boxfill, isofill, isoline, and meshfill; plots for vector data such as vector plots and streamlines; and 3D plots for scalar and vector data. We contributed to the development and maintenance of UV-CDAT but developing a testing infrastructure and solving bugs.

Our other effort includes enhancement to CAM and MPAS I/O and visualization in ParaView. ParaView is an open-source, multi-platform data analysis and visualization application. ParaView users can quickly build visualizations to analyze their data using qualitative and quantitative techniques. The data exploration can be done interactively in 3D or programmatically using ParaView's batch processing capabilities. ParaView was developed to analyze extremely large datasets using distributed memory computing resources. It can be run on supercomputers to analyze datasets of petascale size as well as on laptops for smaller data, has become an integral tool in many national laboratories, universities and industry, and has won several awards related to high performance computation.

Accomplishments

UV-CDAT

1. Buildbot-GitHub integration to use for UV-CDAT testing
   UV-CDAT has used Travis-CI for continuous integration testing for some time. As a free service, Travis-CI limits the total amount of time a build can run. Because the computational resources available on a Travis-CI node is quite small, we have been very limited in the amount of testing that we can do. In addition, we don't have direct access to the machines where the builds take place, so it is often difficult to debug issues that occur. To solve these issues, we have set up and deployed a custom CI testing system that works very similarly to Travis.
   We maintain a Buildbot server locally that listens for change events on the UV-CDAT repository. When a change occurs, the buildbot server will trigger builds on one or more slaves. The build slaves notify Github of the status of the test when the build is finished. The details link goes to the CDash submission for the test, which provides detailed information about what failed and why including image comparisons for tests involving screenshots.


Add more commits by pushing to the vcs3D_fix_dv3d_list branch on UV-CDAT/uvcdat.

✗ Failed — 2 failing and 2 successful checks — Hide all checks

✗ continuous-integration/kitware-buildbot/uvcdat-garant-linux-release/ — Build done. — Details

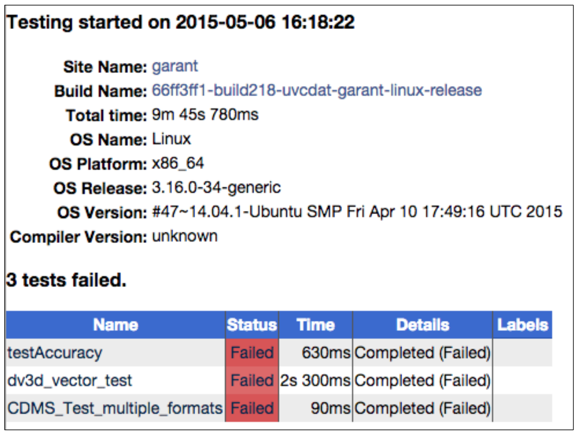
✗ continuous-integration/kitware-buildbot/uvcdat-test-laptop-linux-release/ — Build do... — Details

✓ continuous-integration/travis-ci/push — The Travis CI build passed — Details

✓ continuous-integration/travis-ci/pr — The Travis CI build passed — Details

Merge with caution!
You can also merge branches on the command line.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356

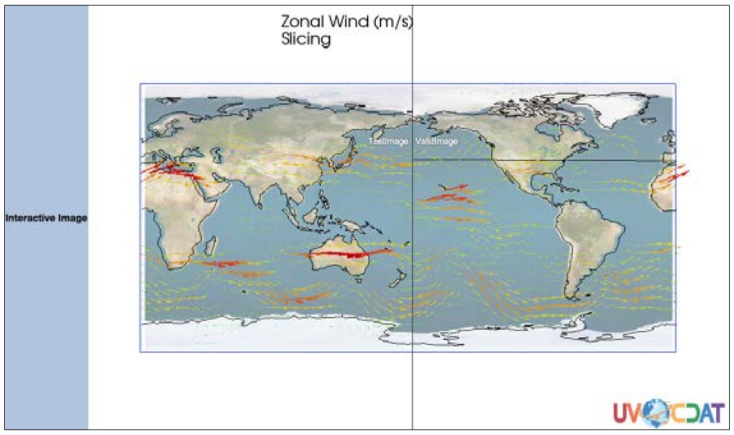


Testing started on 2015-05-06 16:18:22

Site Name:	garant
Build Name:	66ff3ff1-build218-uvcdat-garant-linux-release
Total time:	9m 45s 780ms
OS Name:	Linux
OS Platform:	x86_64
OS Release:	3.16.0-34-generic
OS Version:	#47~14.04.1-Ubuntu SMP Fri Apr 10 17:49:16 UTC 2015
Compiler Version:	unknown

3 tests failed.

Name	Status	Time	Details	Labels
testAccuracy	Failed	630ms	Completed (Failed)	
dv3d_vector_test	Failed	2s 300ms	Completed (Failed)	
CDMS_Test_multiple_formats	Failed	90ms	Completed (Failed)	



2. Fixed bugs and implemented new features in UV-CDAT
    a. BUG #1811: Show point information for plots using a geographic projection. The computation previously used in UV-CDAT was assuming uniform grid so it could not work for data using a geographic projection.

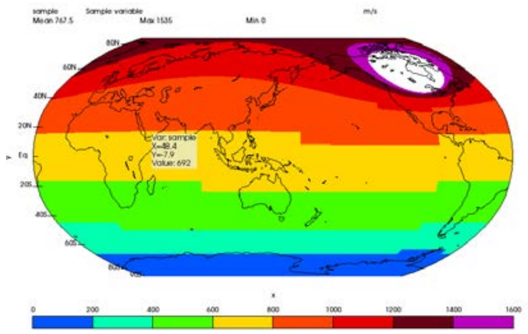


Fig. 1 shows a meshfill plot using Robinson projection. Information such as longitude, latitude and attribute value is shown for the point clicked.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356

b. BUG #1886: Polar projection does not change pole. The order of the Y (latitude) axis specifies the pole that is used for the projection. If Y1 < Y2 we use the South Pole, otherwise we use the North Pole. This specification was not implemented correctly before our fix.
c. ENH #1880: ratio=autot now works for geographic projected datasets. This option instructs UV-CDAT to use the appropriate X/Y ratio for data instead of changing the ratio based on the window size. Fig. 2 shows an example of a plot before our change where the image is deformed based on window X/Y ratio and after our change where the image maintains the data ratio.

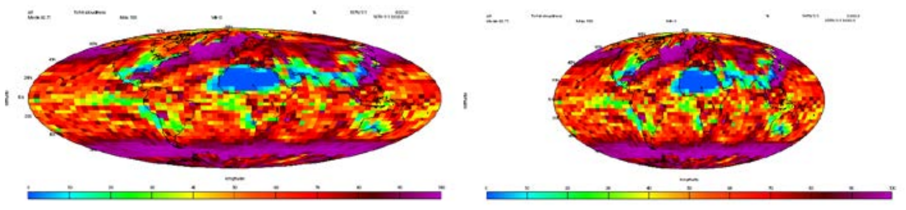
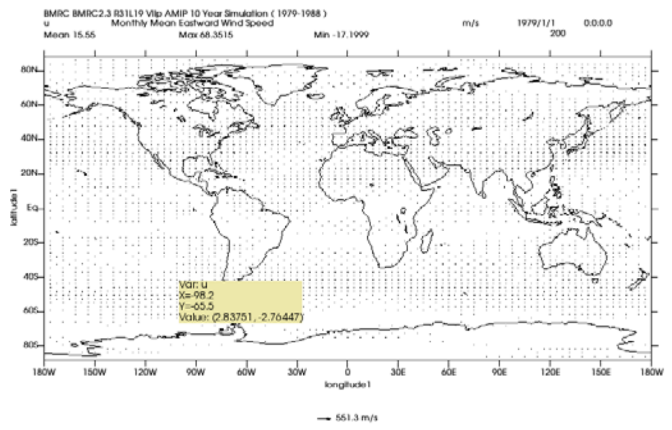


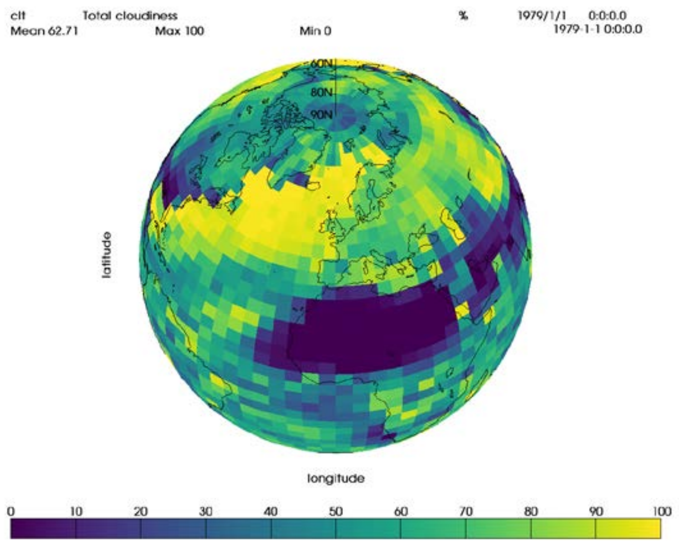
Fig 2. Boxfill with Mollweide projection. Images before (left) and after (right) our fix with the same window size. Current behaviour (right) maintains the data ratio rather than deforming the data to match the window ratio (left).

d. ENH #1881: Add plot_based_dual_grid option to plot(). Traditionally, we created a point or cell dataset based on the plot requested. For isofill, isoline and vector we created point datasets, for boxfill and meshfill we created cell datasets. We keep this behavior for backward compatibility but we add a parameter plot_based_dual_grid to plot(). If this parameter is missing or it is True, we have the traditional behavior. If this parameter is False, we create the dataset that is specified in the file, regardless of the plot requested.
e. ENH #1885: Show info at clicked point for point datasets


BMRC BMRC2.3 R31L19 Vlip AMIP 10 Year Simulation (1979-1988)
u Monthly Mean Eastward Wind Speed m/s 1979/1/1 0.0.0.0
Mean 15.55 Max 68.3515 Min -17.1999 200
Var: u
X=-98.2
Y=-65.5
Value: (2.83751, -2.76447)
latitude1
longitude1
551.3 m/s

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356

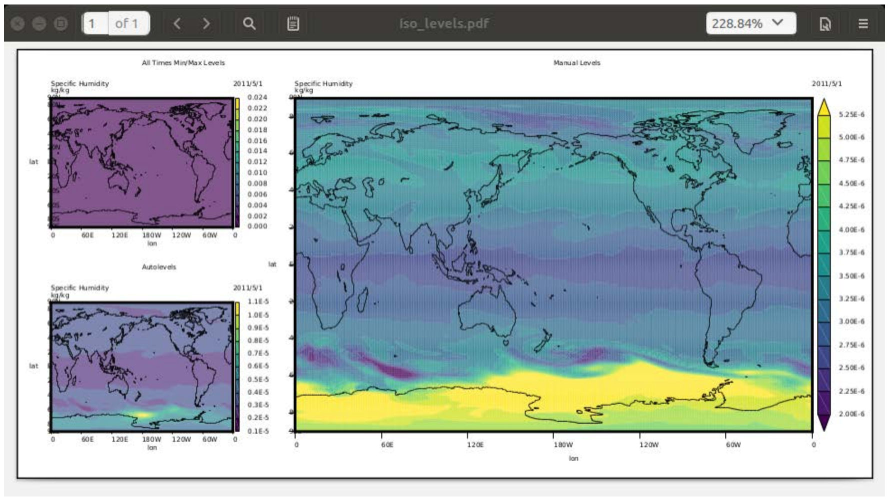
f. BUG #1959: Fix memory override for vtkContourFiler in isofillpipeline. This resulted in vtkStripper to generate double coverage of isocountours which resulted in messed-up patterns. Also adjusted plot patterns to easier to discriminate.
g. BUG #1985: orthographic projection plot is empty. This is because proj4 sets points that are not visible to infinity. We set those points to 0 and hide them.



h. BUG #1947: isofill does not handle out of bounds levels correctly. When smallest level is bigger than min scalar value or highest level is smaller than max scalar value isofill creates the wrong image. Also, out of range (white color) was shown black.
i. BUG #1944: Rename line to linetype for isoline, unified1d and vector.
j. BUG #16744 in VTK: Resources are not removed for a renderer removed from the window.
k. BUG #1770: Display meshfill template elements through renderTemplate. This deletes some displays which reduces memory leaks.
l. BUG: Fix system dependent display of the outline of a dataset. We increase the parallel projection parallelepiped with 1/1000 so that it does not overlap with the outline of the dataset. This resulted in system dependent display of the outline.
m. ENH in VTK: Add vtkTextProperty::UseTightBoundingBox to center a label to anchor. This property allows the user to center the string to the anchor point. By default, the bounding box for the string is computed using the font metrics which includes ascents and descents. As descents might not be present in the string, the label will not be perfectly centered on the anchor point. Setting this property on will compute the bounding box using the current string rather than the font metric. This results in perfectly centered labels. This does not work well for temporal data if the string changes as it results in text that moves around.
n. BUG in VTK: Improve text alignment.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356

o. BUG: Fix isofill levels: BUG #1265: Fix datawc zoom-in for geographic projections, BUG #5: vtkPolyData::RemoveDeletedCells does not remove GlobalIds or points, We set scalar value for hidden points to a valid value. BUG: Set hidden points scalars and vectors to minimum value from the range.
p. BUG #16909 in VTK: Save vtkLogoRepresentation to postscript.
q. ENH: VCS export to matplotlib

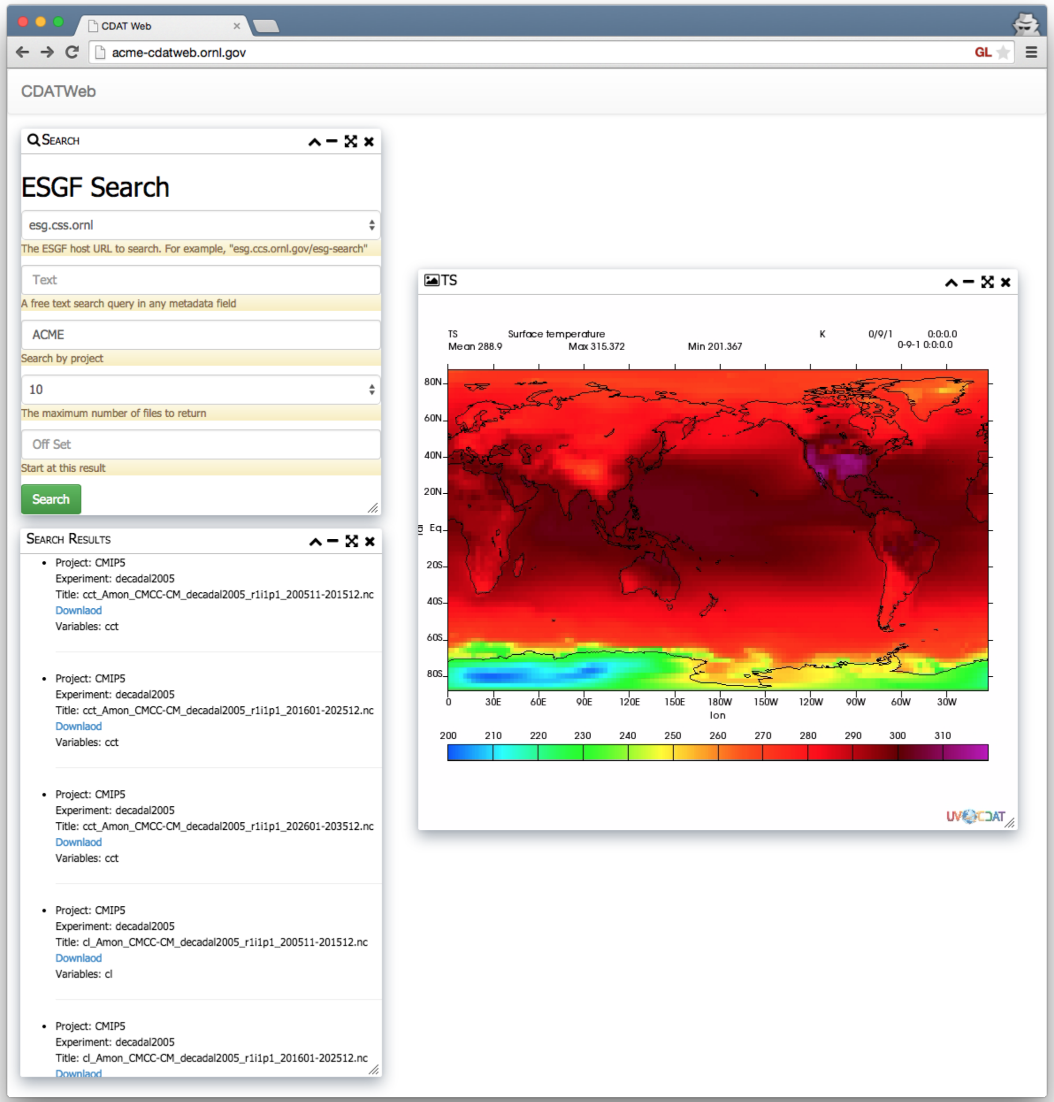


r. Worked on Virtual GL + Turbo VNC installation to support UV-CDAT customers that intend to plot datasets on remote machines.

vCDAT

1. vCDAT is developed such that it cleanly separates the visualization server (UV-CDAT visualization and analysis using vtk-web) from the web server (Django application). As a part of this refactor, we set up a series of Bootstrap templates from which to derive all of the web components for ACME to maintain a unified appearance. The templates are also designed so that the components of vCDAT can be used as part of the overarching dashboard user interface being developed in parallel.
   a. vtk-web launcher
   In order to serve the application as a service, it is necessary to have the ability to launch new visualization instances on demand. The vtk-web launcher is designed to exist as an internal service that the main Django application can post requests to for a logged in user. The launcher is responsible for starting up a python process that will communicate with the user's web browser over a websocket. For security purposes, the launcher itself nor any of the visualization nodes should be accessible from the outside. Instead, the launcher configures the Apache instance running the application to proxy websockets to the appropriate host and port. The launcher also handles closing old instances and spreading the load among the available resources.
   b. Visualization server
   The visualization server itself uses the web RPC library called Autobahn by wrapping python functions that make use of UV-CDAT to create visualizations and stream the results back to the client as images. The Autobahn library makes it easy to expose new functionality from UV-CDAT as development continues.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356



ESGF Search

esg.css.ornl
The ESGF host URL to search. For example, "esg.ccs.ornl.gov/esg-search"

Text
A free text search query in any metadata field

ACME
Search by project

10
The maximum number of files to return

Off Set
Start at this result

Search

Search Results

- Project: CMIP5
  Experiment: decadal2005
  Title: cct_Amon_CMCC-CM_decadal2005_r1i1p1_200511-201512.nc
  Download
  Variables: cct
- Project: CMIP5
  Experiment: decadal2005
  Title: cct_Amon_CMCC-CM_decadal2005_r1i1p1_201601-202512.nc
  Download
  Variables: cct
- Project: CMIP5
  Experiment: decadal2005
  Title: cct_Amon_CMCC-CM_decadal2005_r1i1p1_202601-203512.nc
  Download
  Variables: cct
- Project: CMIP5
  Experiment: decadal2005
  Title: cl_Amon_CMCC-CM_decadal2005_r1i1p1_200511-201512.nc
  Download
  Variables: cl
- Project: CMIP5
  Experiment: decadal2005
  Title: cl_Amon_CMCC-CM_decadal2005_r1i1p1_201601-202512.nc
  Download

TS Surface temperature K 0/9/1 0:0:0.0
Mean 288.9 Max 315.372 Min 201.367 0-9-1 0:0:0.0

c. Server deployments
As a proof of concept, all components of the CDATWeb system have been deployed on two virtual machines at ORNL. The Django front end is deployed at acme-cdatweb.ornl.gov and the visualization launcher at acme-uvcdat.ornl.gov. These deployments will continue to be updated as development of vCDAT proceeds to collect feedback on the user interface. Another development deployment of the visualization server has been deployed at aims1.llnl.gov which is intended to be used for internal development and testing.

2. Continuous integration testing infrastructure was added to vCDAT's github repository. On push to the repository a job is submitted to CircleCI running the test suite and reporting back to github's status API. For the initial infrastructure, both client side (jshint, jscs) and server (flake8) linting and style tests were added.
3. Additional development includes:

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356

a. UV-CDAT docker builds
   One significant problem encountered while developing CDATWeb was that the visualization server requires a full install of UV-CDAT to work on. To aid other developers, we have created a docker image containing a bleeding edge version of UV-CDAT available at docker hub (https://registry.hub.docker.com/u/uvcdat/uvcdat/). This image makes it easy to try out command line scripts for UV-CDAT. For example once docker is installed, a user can execute a UV-CDAT script as follows:

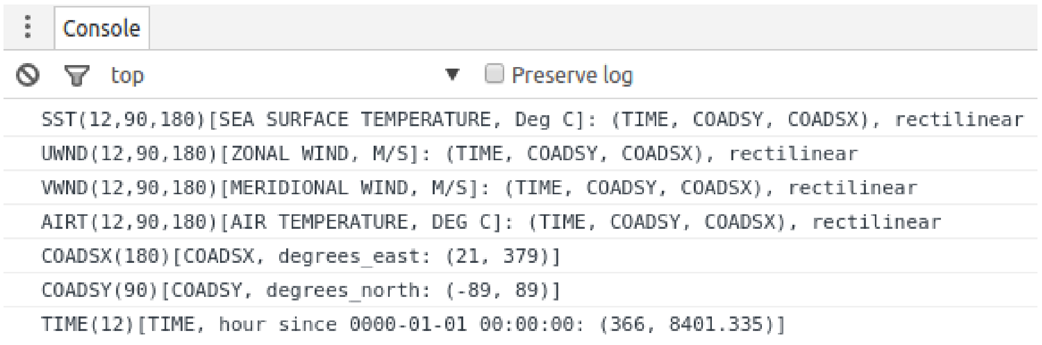
   ```
   docker run -t uvcdat/uvcdat script.py
   ```

   The base UV-CDAT docker image was extended to support launching visualization servers in a similar manner:

   ```
   docker run -p 8000 -t uvcdat/cdatweb-vtkweb
   ```

   This method of launching a visualization server can be used with the vtk-web launcher to simulate the functionality of a real deployment. Documentation in the vCDAT repository describe how to configure the launcher for local testing.
b. We developed vagrant and ansible provisioning for UV-CDAT.
   Ansible provisioning playbooks and vagrant configurations were added to the UV-CDAT repository to quickly spin up a repeatable development environment. The deployment scripts feature a configuration file to enable optional features in the provisioned virtual machine including memory allocation and GUI support.
c. We investigated ESGF authentication. Redeployments of the ESGF servers has broken the ability to log in to the ESGF through UV-CDAT. Because most of the data hosted on the ESGF requires an authenticated connection to be accessed, this was a major blocker for the vCDAT ESGF interface. Kitware investigated this issue and concluded that it was likely a server problem (https://github.com/UV-CDAT/uvcdat/issues/1337).
d. We listed plot type and templates in vCDAT client. To help in the development of the client side user interface, RPC methods were added to server interface that will list all available plot types and templates. This is used by the client to render a list of choices that the user can select.
e. We fixed a vtk-web bug preventing multiple plot windows to be created reliably. Multiple plot windows could not reliably be opened in vCDAT because of a bug in VTK related to comparing pointers. This was fixed in pull request to VTK (https://github.com/UV-CDAT/VTK/pull/1).
f. We returned full variable information to the web client. This information enables the client to create a variable selection and subsetting dialog similar with the Load Variable dialog in the Qt UV-CDAT interface. This dialog lists all variables present in a NetCDF file. For each variable, the dialog lists all its axes, and allows the user to specify a new range for each axis. This way, the user can load the entire variable or, only a subset of it. Fig. 3 shows the information sent to vCDAT client for a NetCDF file.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356



```
SST(12,90,180)[SEA SURFACE TEMPERATURE, Deg C]: (TIME, COADSY, COADSX), rectilinear
UWND(12,90,180)[ZONAL WIND, M/S]: (TIME, COADSY, COADSX), rectilinear
VWND(12,90,180)[MERIDIONAL WIND, M/S]: (TIME, COADSY, COADSX), rectilinear
AIRT(12,90,180)[AIR TEMPERATURE, DEG C]: (TIME, COADSY, COADSX), rectilinear
COADSX(180)[COADSX, degrees_east: (21, 379)]
COADSY(90)[COADSY, degrees_north: (-89, 89)]
TIME(12)[TIME, hour since 0000-01-01 00:00:00: (366, 8401.335)]
```

Fig. 3. Information sent to vCDAT client about variables in a NetCDF file. For instance SST is a variable stored on a rectilinear grid with extents TIME=12, COADSY=90 and COADSX=180.

g. We added the ability to create plots based on subsetted variables. This is done by including additional specification when creating the plot: for each variable used in the plot, we can specify a list of axes name and range pair. Fig. 4 shows a vector plot for a full grid and for a portion of it.

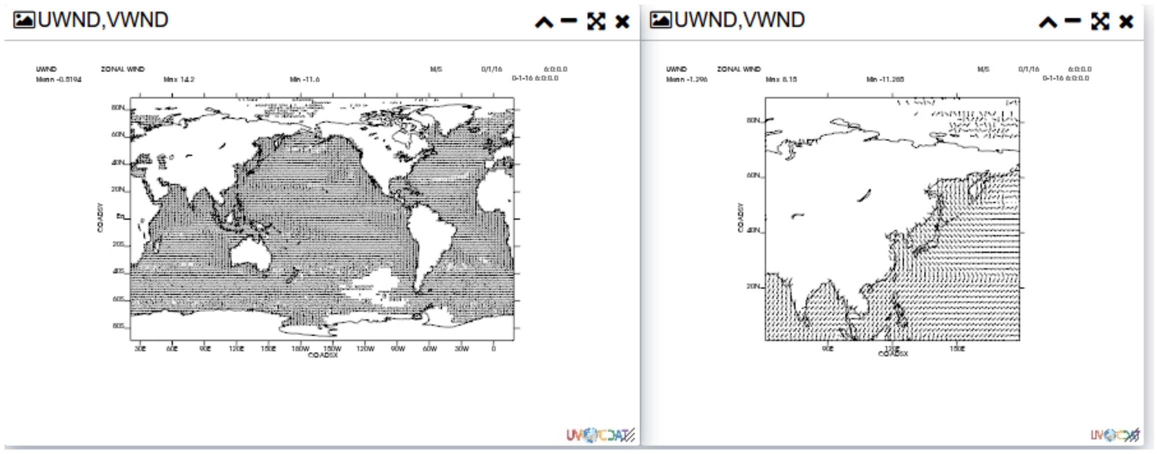
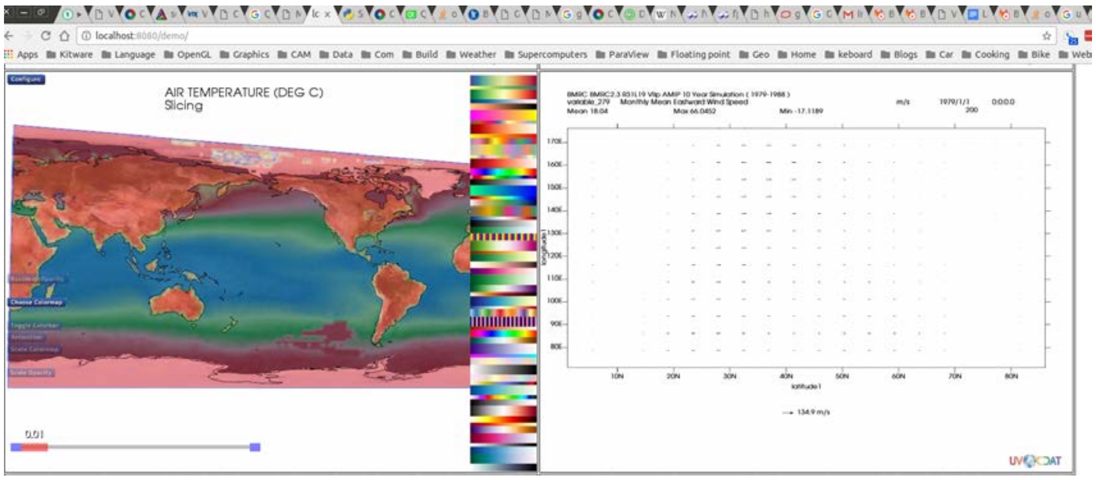


Fig. 4. Vector plot for a grid with longitude between 21 and 379 and latitude between -89 and 89 (left) and a vector plot for the same grid only showing longitude between 60 and 180 and latitude between 0 and 90.

h. Defined VCS.js, an API and a library for the accessing web based, VCS like plotting facility. VCS.js defines both server and client based plotting. vCDAT is using VCS.js as a result of this work. See https://github.com/UV-CDAT/vcs-js for the implementation of this library.
i. vCdat experimented to using plain RPC for implementing the server side calls to VCS causing major performance issues. We switched back to using vtk-web for showing VCS plots in the browser which fixed the performance issues experienced. We support clicking on a plot to show context information,

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356

interaction with 3D plots, multiple plots in different canvases or on the same canvas and we partially support resizing of a plot. In the next figure we show a 3D plot and a vector plot of a subset variable. The following figure shows a vcs 3D plot and a vector plot in a browser window.



ParaView

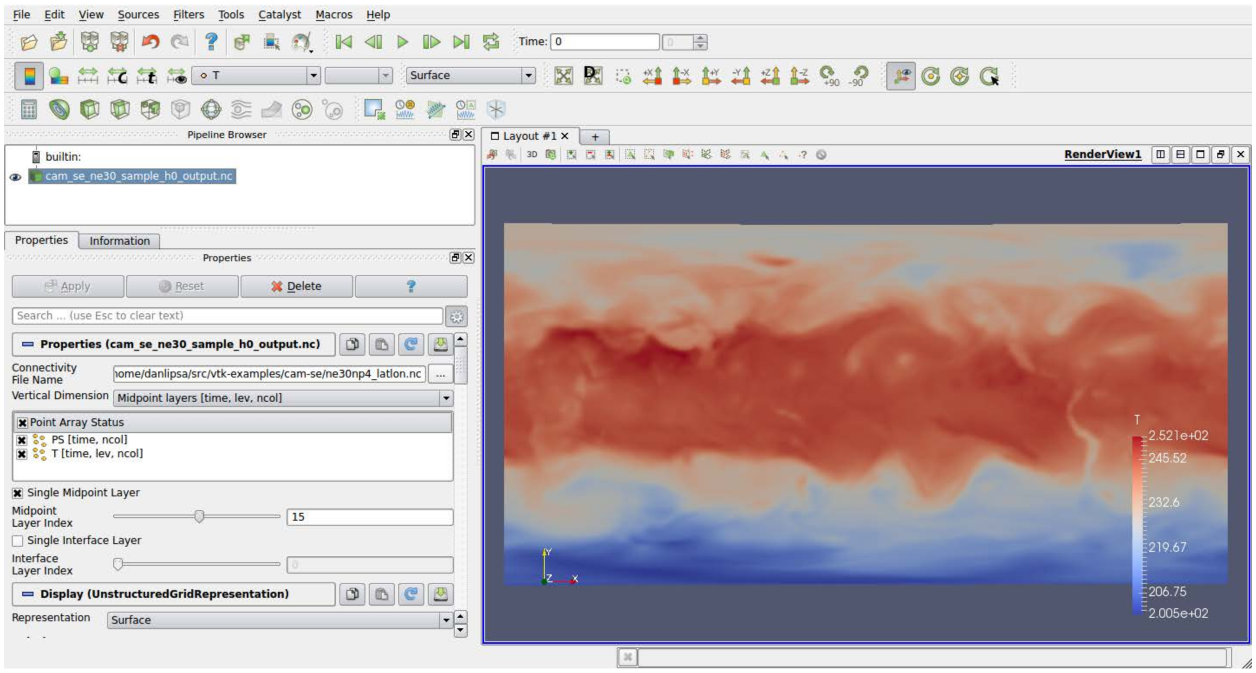
CAM NetCDF Reader

The CAM NetCDF Reader reads in files produced by the Community Atmospheric Model (CAM) simulation program and produces an unstructured grid.
The grid is unstructured in the XY plane but it is rectilinear in the Z direction. This results in layers of hexahedral cells along the Z direction. If we read only one layer we produce quadrilateral cells.

There are three kind of attributes in these files: Single layer attributes, midpoint layer attributes and interface layer attributes. Single layer attributes are associated with the XY point data and depend on variables (time, ncol). Midpoint layer attributes are associated with the XY points and the midpoint of a layer. These attributes depend of variables (time, lev, ncol). Interface layer attributes are associated with the XY points and the interface points of layers. These attributes depend on variables (time, ilev, ncol). Note that the number of interface layers (ilev) is equal with the number of midpoint layers (lev) plus one.
The reader requires two files: the main file has all attributes, the connectivity file has point positions and cell connectivity. Improvements include

1. Select to read single layer, midpoint layer and interface layer attributes using the Vertical Dimensions reader setting. A user could read only single layer and midpoint layer attributes before this work.
2. Specify which attributes are read from the file. This feature saves time and memory as only the data needed is loaded.
3. Choose to read only a single layer instead of reading all layers. This feature saves time and memory and enables further analysis of individual data layers.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356



builtin:
cam_se_ne30_sample_h0_output.nc

Properties (cam_se_ne30_sample_h0_output.nc)
Connectivity File Name: home/danlipsa/src/vtk-examples/cam-se/ne30np4_latlon.nc
Vertical Dimension: Midpoint layers [time, lev, ncol]
[✓] Point Array Status
[✓] PS [time, ncol]
[✓] T [time, lev, ncol]
[✓] Single Midpoint Layer
Midpoint Layer Index: 15
[ ] Single Interface Layer
Interface Layer Index: 0
Display (UnstructuredGridRepresentation)
Representation: Surface

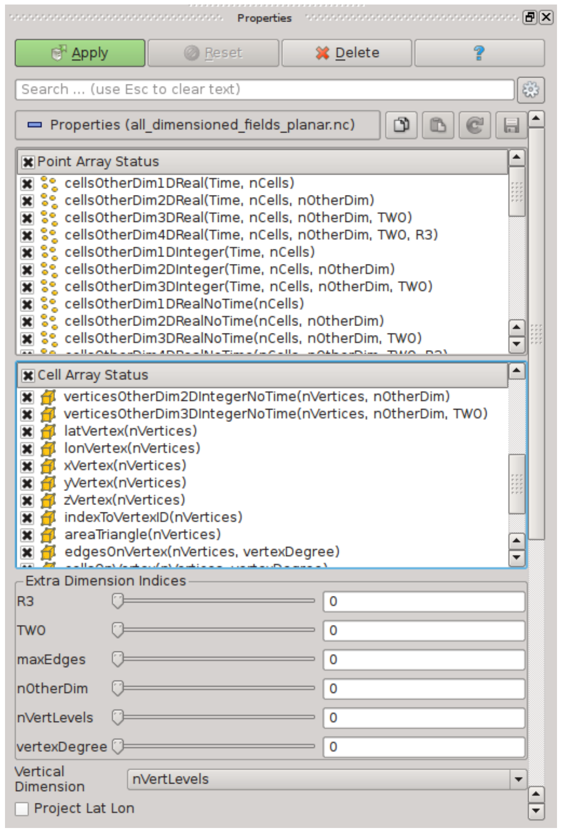
MPAS NetCDF Reader

We provide a number of improvements to the MPAS NetCDF Reader such are reading new attributes invisible before, better using the VTK pipeline to improve performance and cleaning up the code. UI changes include adding an Extra Dimension Indices

- Support for loading arrays with data types other than double.
- Removed limit of 100 attribute arrays.
- Old version only read attribute arrays with dimensions:
  ([Time,] (nCells | nVertices) [, nVertLevels)
  New version will load arrays with dimension signature:
  ([Time,] (nCells | nVertices), [dim1, [dim2, [dim3, ...]]])
  Vertical dimension is configurable at runtime. Defaults to nVertLevels for backwards compatibility. Other dimensions (dim1, dim2, etc above) are fixed at a configurable value.
- A vertical dimensions is no longer required to read an MPAS NetCDF file.
- Array names now optionally include dimension info. Instead of just, e.g. “tracers”, users will see "tracers(Time, nCells, nVertLevels, nTracers)". This feature can be disabled via the UseDimensionedArrayNames boolean.
- Removed special loading of “vertexMask” array. This was hardcoded to expect a certain signature (which wasn’t always used), and crashed if the file didn’t match. The array is now loaded using the generic attribute array mechanism.
- Added mechanism to validated dimensions of NetCDF variables before loading them, improving general stability.
- Refactored reader to properly use pipeline. Some Set[...] methods were causing the data to be regenerated immediately, rather than defering to RequestData.
- Added awareness of the “on_a_sphere” attribute, and updated the EliminateXWrap method and multilayer extrusion code appropriately for when the data is planar.
- Removed many unused variables and unimplemented methods from the vtkMPASReader API.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356

Changes to the ParaView UI



Properties

Apply | Reset | Delete | ?

Search ... (use Esc to clear text)

Properties (all_dimensioned_fields_planar.nc)

- [x] Point Array Status
  - [x] cellsOtherDim1DReal(Time, nCells)
  - [x] cellsOtherDim2DReal(Time, nCells, nOtherDim)
  - [x] cellsOtherDim3DReal(Time, nCells, nOtherDim, TWO)
  - [x] cellsOtherDim4DReal(Time, nCells, nOtherDim, TWO, R3)
  - [x] cellsOtherDim1DInteger(Time, nCells)
  - [x] cellsOtherDim2DInteger(Time, nCells, nOtherDim)
  - [x] cellsOtherDim3DInteger(Time, nCells, nOtherDim, TWO)
  - [x] cellsOtherDim1DRealNoTime(nCells)
  - [x] cellsOtherDim2DRealNoTime(nCells, nOtherDim)
  - [x] cellsOtherDim3DRealNoTime(nCells, nOtherDim, TWO)
- [x] Cell Array Status
  - [x] verticesOtherDim2DIntegerNoTime(nVertices, nOtherDim)
  - [x] verticesOtherDim3DIntegerNoTime(nVertices, nOtherDim, TWO)
  - [x] latVertex(nVertices)
  - [x] lonVertex(nVertices)
  - [x] xVertex(nVertices)
  - [x] yVertex(nVertices)
  - [x] zVertex(nVertices)
  - [x] indexToVertexID(nVertices)
  - [x] areaTriangle(nVertices)
  - [x] edgesOnVertex(nVertices, vertexDegree)

Extra Dimension Indices

R3	0
TWO	0
maxEdges	0
nOtherDim	0
nVertLevels	0
vertexDegree	0

Vertical Dimension: nVertLevels

- [ ] Project Lat Lon

- Display new dimensioned array names. All dimensions other than “Time”, “nCells”, and “nVertices” appear in a new “Extra Dimensions Indices” box, where the slice index can be set. The vertical dimensions can be selected from a dropdown box of all “special” dimensions. Disabled when MultiLayer is not enabled.
- Removed the existing “Vertical Level” slider, as it’s redundant with the new “Extra Dimensions” UI.
- Updated the remove-periodic.cpd filter to reflect recent changes in ParaView (numpy must be imported manually post-4.0).

Project Impact

Scientific visualization plays an important role in current scientific endeavors. Visualization transforms abstract data into images on a computer screen. Scientists examine and interact with these representations of the data to gain scientific insight and better understanding of the phenomenon studied.
We work on two products that deliver visualization to scientists involved in climate research: UV-CDAT - a visualization library and desktop application for climate data and vCDAT - a web based visualization application which uses UV-DAT at its core. We improved and developed UV-CDAT in important ways which include:

- We enable running UV-CDAT in a reproducible way using docker as well as running it in a virtual machine using vagrant and ansible. These tools can save valuable time in deploying UV-CDAT on a target platform. This is important for scientists as well as vCDAT developers.
- We improved the interactive query tool which enable scientists to get information about any point in the visualization improving their analysis capabilities.

- We improved data presentation for data transformed through geographic projections. This enables scientists to see data as is rather than being deformed based on the window size.

Improvements in UV-CDAT have an equal impact in vCDAT, as this web-based tool has UV-CDAT at its core. For vCDAT we continued development efforts and moved closer to having a tool that delivers similar capabilities with the UV-CDAT desktop visualization application.

- We deliver plot types and templates to the client. This enables scientists to interactively choose the desired visualization to match their scientific goals.
- We deliver full data information to the client. This enables the web client to build an interface to interactively load the data to visualize.
- We enabled visualizing only a section of the full data. Together with the previous bulled this enables scientists to zoom-in and analyze only the section of interest rather than the full data.

Climate scientists use our tools to visualize data. Scientists explore and analyze climate data which enables new insights and new understanding of the science behind the studied phenomenon. This ultimately leads to improvements in climate models. Visualization is also used in communicating scientific results which results in increased collaboration which is one of the catalysts of scientific discovery.

Kitware Contributors

- Aashish Chaudhary, Principal Investigator: Managed project and served as the technical lead for development efforts.
- Jonathan Beezley, Research and Development Engineer: Developer for vCDAT and UV-CDAT.
- Dan Lipsa, Research and Development Engineer: Developer for UV-CDAT and vCDAT.
- Allisson Vacanti, Research and Development Engineer: Developer for UV-CDAT.
- Sankhesh Jhaveri, Research and Development Engineer: Developer for UV-CDAT.

Training and Professional Development

Weekly meetings were held. Most weeks, a different member of team presents a paper or a technology of interest to our work in the project. This allows us to learn about new ideas and technologies and discuss possible applications in our work. We continue to use web resources to improve our skills in visualization, server side and web development.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Award Number DE-SC0012356