J3.22

THE PCMDI CLIMATE DATA ANALYSIS TOOLS (CDAT) – AN OPEN SYSTEM APPROACH TO THE IMPLEMENTATION OF A MODEL DIAGNOSIS INFRASTRUCTURE

Michael Fiorino[1] (fiorin@llnl.gov) and Dean Williams (williams13@llnl.gov)

Lawrence Livermore National Laboratory
Program for Climate Model Diagnosis and Intercomparison
Livermore, CA 94550
Software Web Site: http://cdat.sourceforge.net

1. INTRODUCTION

The Climate Data Analysis Tools (CDAT) is a set of separate software subsystems that are glued together using the object-oriented Python scripting language (http://www.python.org) to form an infrastructure for model diagnosis and other data analysis tasks. The power of the system comes from Python and the software subsystems. Python provides a general purpose and full-featured scripting language with a variety of user interfaces including command line interaction, stand-alone scripts (applications) and GUIs. The CDAT subsystems, implemented as Python modules, provide access and management of gridded data (Climate Data Management System or CDMS); large-array numerical operations (Numpy Mask Arrays or MA); and visualization (Visualization and Control System or VCS).

We characterize CDAT as "open system" because the software subsystems are independent and the object-oriented nature of Python allows CDAT to be "delay bound" or that the actual tool is built at run time, i.e., is not fixed. Further, the software subsystem (Python "modules") are open source and freely available for community-wide development. Thus, CDAT is easily extended and represents a different approach to the technical problem of implementing a model diagnosis infrastructure.

In this paper, we compare and contrast the CDAT approach with more traditional tools built from system-level software (e.g., C and X windows), such as the Grid Analysis and Display System GrADS (http://grads.iges.org/grads) and FERRET (http://ferret.wrc.noaa.gov/Ferret/), and show how CDAT complements and offers an alternative interface to data accessible by these popular tools. We also demonstrate CDAT-hosted data service applications using the popular Live Access Server (LAS) and Distributed Oceanographic Data System (DODS) systems and using the metadata search capabilities of CDMS.

2.0 CDAT COMPONENT SUBSYSTEMS

The key CDAT subsystem is CDMS as it provides access to gridded data in variety of formats and higher-level organization via an XML representation of sets of data. The "cdunif" data interface of CDMS is a unified application programmer interface (API) containing the I/O layers of: 1) GrADS (binary floats and integers and WMOGRIB formats); and 2) netCDF (COARDS and CF compliant (http://www.cgd.ucar.edu/cms/eaton/netcdf/CF-current.htm). The cdunif layer has been successfully tested with HDF and other formats, but the GrADS/netCDF interface supports the primary data sets used in climate model research.

Python has attracted a large following in the scientific community because of the Numpy module (http://www.pfdubois.com/numpy/). This module features a comprehensive collection of array processing functions and has been extended to handle the common situation in gridded data where some points have missing values (the "MA" module). The performance of Numpy is excellent because the calculations are C-code based.

CDAT provides interfaces to two graphics systems for display of data objects: 1) VCS, specialized for 2-D field plotting and animation; and 2) GRACE (http://www.math.nyu.edu/AML/software/xmgrace.html) for 1-D line plots. Python allows relatively straightforward interfaces to other systems and is one of the key benefits of basing CDAT on a glue or scripting language.

3.0 CDAT CONCEPT OF OPERATION

The basic function of CDAT is the creation and manipulation of data objects. These data objects consist of data (metadata and gridded fields in an N-dimensional array) and methods (data object functions). The first step is making a Python application that "imports" Python modules (objects with data and functions) for processing such as CDMS for reading data sets and MA for performing numerical computations. Because CDAT data objects are Pythonic (i.e., Python object), they have a consistent structure that is understood by other Python-hosted modules. Thus, CDAT objects have access to wide range of standard Python modules, from database to Internet protocols. Further, CDAT applications are flexible as they depend only on the modules imported and are created only when executed. In contrast, the GrADS and FERRET applications are fixed and their data objects are internal and specific with limited access to external processes.

A typical CDAT data application would import CDMS, open a data set, extract data (create an object), perform an analysis function (transform the object), and then pass the object to a display object (e.g., VCS) for plotting. Beyond elementary

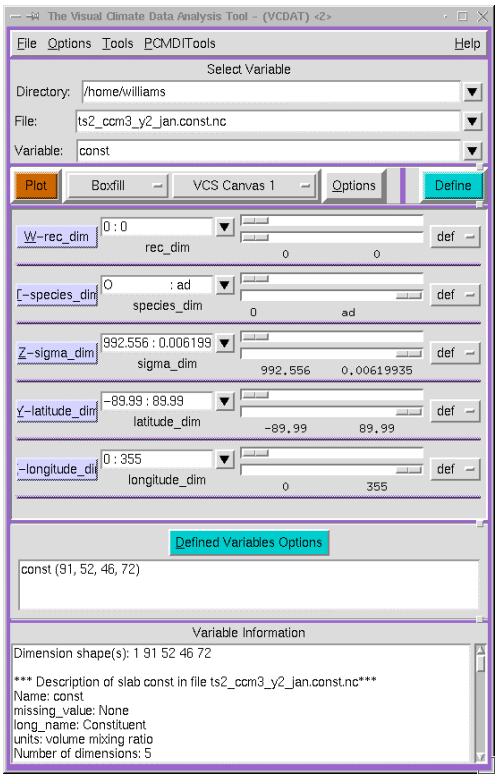
[1] Dr. Michael Fiorino, PCMDI, Lawrence Livermore National Lab, P.O. Box 808, Mail Stop-264, Livermore, CA 94550
UCRL-JC-142879-ABS

modules for data access and calculation, the CDAT system comes with special purpose modules for performing common data analysis tasks with gridded data such as “selectors” to simplify extraction of data in world (e.g., longitude and time) coordinates and averaging in space and time (e.g., calculate DJF seasonal mean). CDAT also features more advanced and complex diagnostic modules including the NCAR spherepak, regridpak routines.

4.0 CDAT USER INTERFACES

The primary user interface is the Python command line either interactively or through a script. We have also implemented a visual interface to CDAT (VCDAT) using Tkinter (http://www.python.org/topics/tkinter/) a thin Tcl/Tk (http://www.sco.com/Technology/tcl/Tcl.html) client.

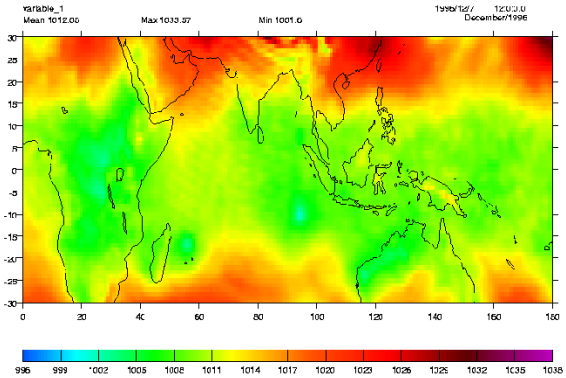
Samples of the interfaces are given below. The first shows the VCDAT GUI:



The second is a script application that demonstrates extraction or slicing of a 4-D variable using world coordinates and plotting:

```
#!/usr/bin/env python
import cdms
# open a GrADS data set
f=cdms.open('model.ctl')
# list variables
f.listvariables()
# create a data object or variable
# sea level pressure in Pa
p=f('psl',time=0,lat=(-30,30),lon=(0,180))
# import vcs and display
import vcs
x=vcs.init()            # create plot window
x.plot(p/100)           # plot in hPa
x.gif('/tmp/t.gif')     # save as gif image
```

The script produces the following plot:



The example above shows that the syntax of data object creation is quite simple and comparable to fixed systems such as GrADS/FERRET. However, unlike fixed system, CDAT gives the user full access to the operating system and other modules through Python. Thus, we believe our system allows for the implementation of a more comprehensive and open infrastructure for gridded data diagnostics.

Future plans include server-side data analysis where a CDAT client requests a calculation (e.g., global average) on a data server running a CDAT server. The availability of Python modules for various Internet and client-server protocols will allow for a more rapid and robust implementation. We will also support more general grids (e.g., non-rectilinear) and build interfaces for observational (point) data.

5.0 RESOURCES

The following links give starting points for CDAT and Python:

- CDAT Project Home Page and Docs: http://cdat.sourceforge.net
- Python Links for New Users: http://www.python.org/doc/Newbies.html
- CDAT Source and Binary Distributions: http://sourceforge.net/project/showfiles.php?group_id=11356

6.0 ACKNOWLEDGEMENTS

CDAT is the collective work of the following team members at PCMDI: Krishna AchutaRao, Charles Doutriaux, Robert Drach, Mike Fiorino, Charlie O’Connor and Dean Williams. This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.