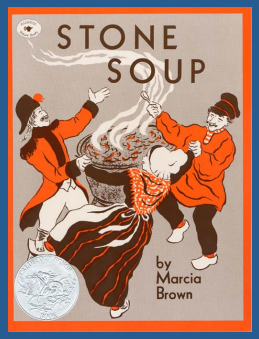
CSDMS Developer Clinic: New Tools and Information for Code Contributors

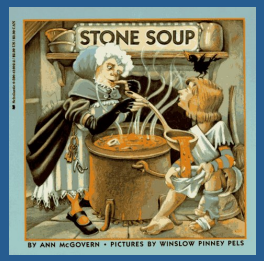
Scott D. Peckham

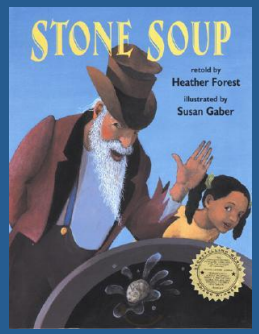
Chief Software Architect for CSDMS

October 15, 2010

One Super Model or Stone Soup?







Where Are We Now ?

- Number of members = 458
- Number of contributed models = 186

  94 Terrestrial, 58 Coastal, 27 Marine,

  50 Hydrology, 4 Carbonate (a few in multiple groups)
- Number of CSDMS components = 49

  hydrology (many), glacier, stratigraphy (many), ocean,

  landscape & coastal evolution, GP eBook, data access
- Frequent, heavy-use of our HPC cluster
- Release of our CMT tool in July 2010
- A grad-student course that uses CMT

Talk Outline

- Developer overview of the CMT
- Guidelines for component development
- Developer overview of netCDF and VisIt
- New CSDMS tools for writing netCDF files
- Developer resources available on the CSDMS cluster
- Creating tabbed-dialog GUIs via XML
- New spatial regridding tools: OpenMI & ESMF
- Role of Python in CSDMS
- Components that access web services
  (e.g. CUAHSI HIS, NED and OpenDAP)
- Future directions

Objectives

The purpose of this clinic is to provide model developers with information and tools that are aimed at:

(1) making their model reusable in a wider variety of contexts

(2) adding new capabilities to their model (e.g. GUI, help files, netCDF output) with the least amount of effort.

In addition, this clinic provides an opportunity to discuss technical issues with the CSDMS Development Team and to provide feedback and suggestions that can help us to improve our modeling system.

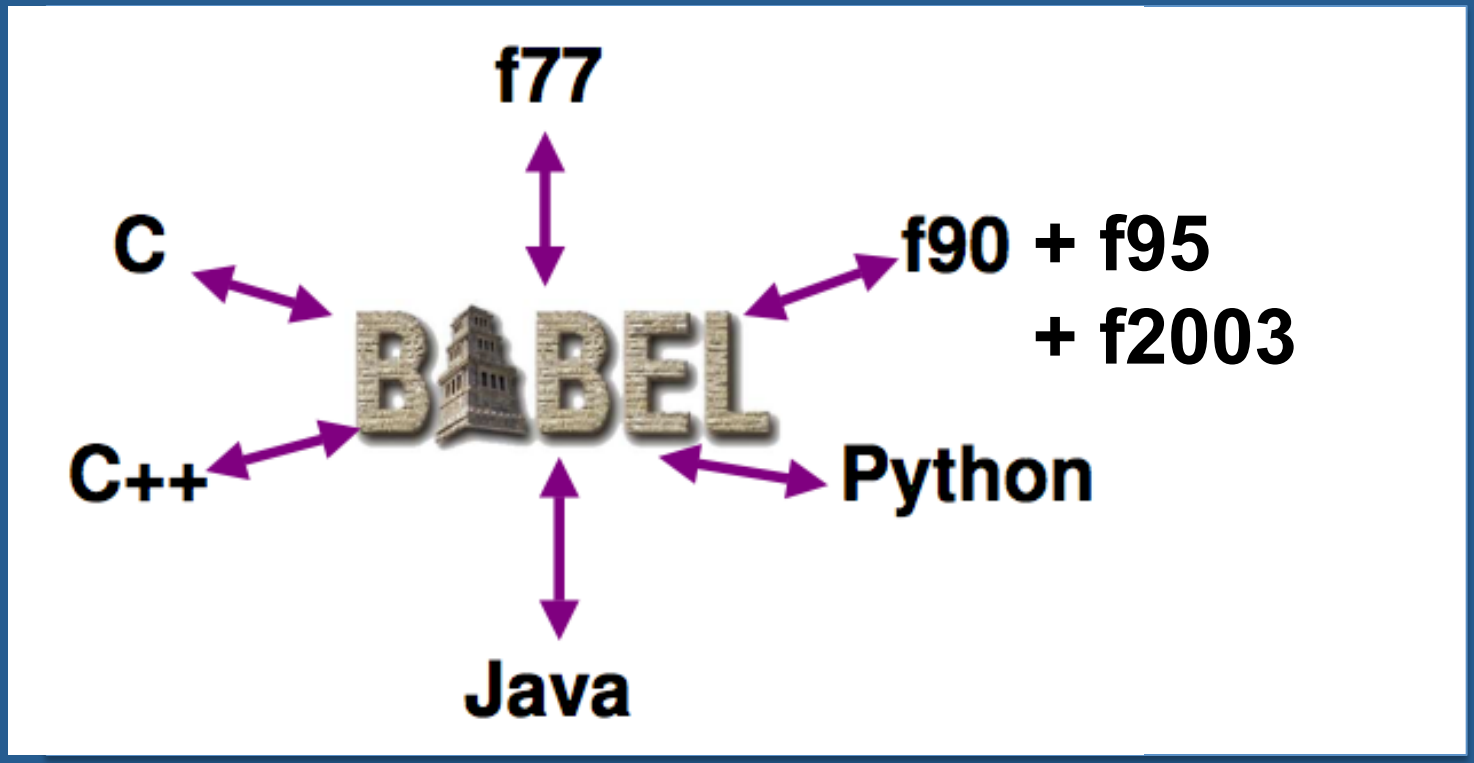
Building a Modeling Framework

CSDMS has integrated a variety of powerful, open-source tools to build its modeling framework, such as:

Babel – Language interoperability (C,C++,Java,Python,Fortran)
Bocca – Component preparation and project management
Ccaffeine – Low-level model coupling (parallel environ.)
ESMF Regrid – Multi-processor spatial regridding
OpenMI Regrid – Single-processor spatial regridding
OpenMI – Component interface standard (1.4 and 2.0)
NetCDF – Scientific data format (self-describing, etc.)
VisIt – Visualization of large data sets (multi-proc.)

We greatly extended the original *Ccaffeine GUI* to create our ***CSDMS Modeling Tool*** for interactive model coupling.

CCA: The Babel Tool

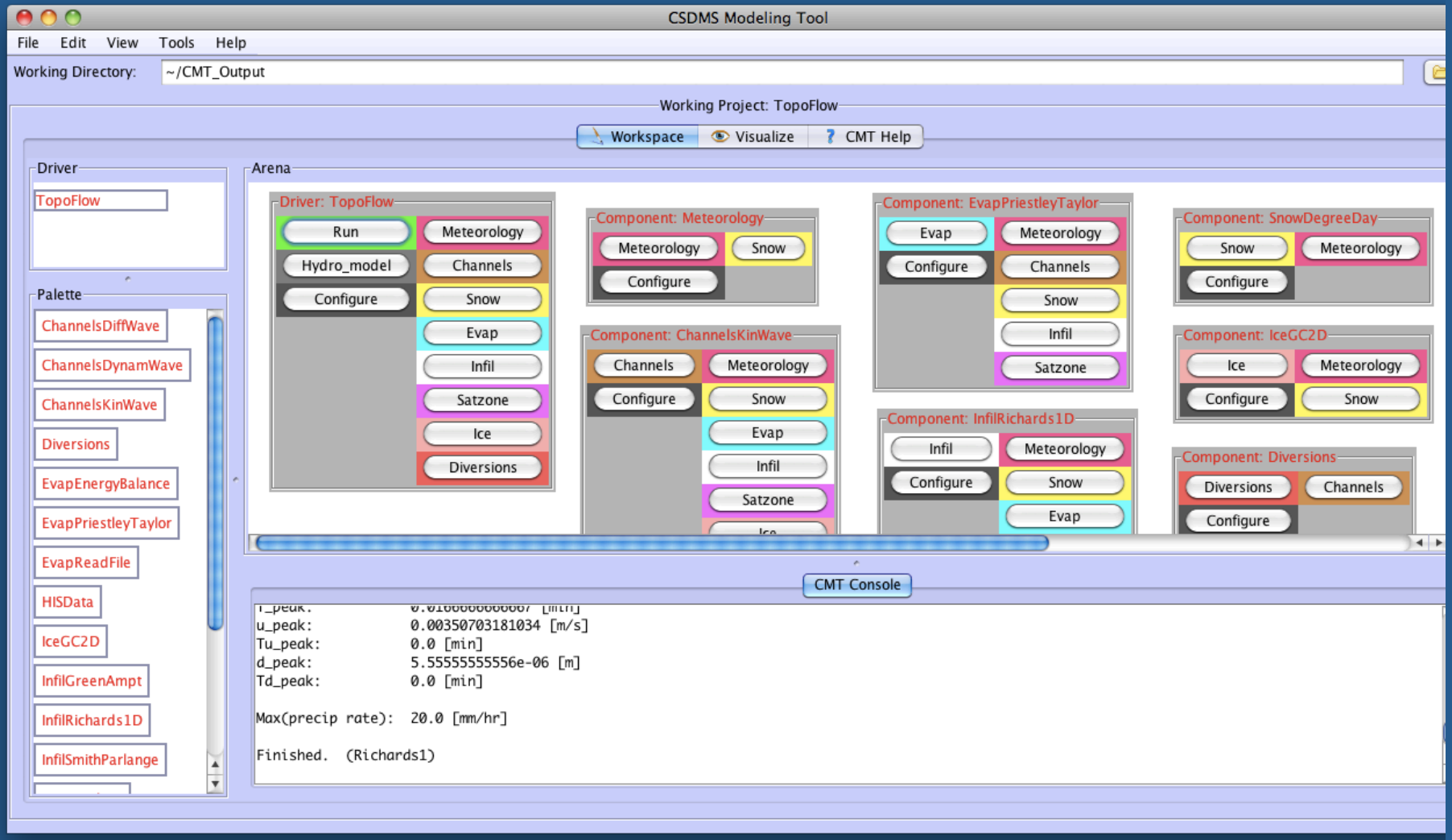


Language interoperability is a powerful feature of the CCA framework. Components written in different languages can be rapidly linked in HPC applications with hardly any performance cost. This allows us to "shop" for open-source solutions (e.g. libraries), gives us access to both procedural and object-oriented strategies (legacy and modern code), and allows us to add graphics & GUIs at will.

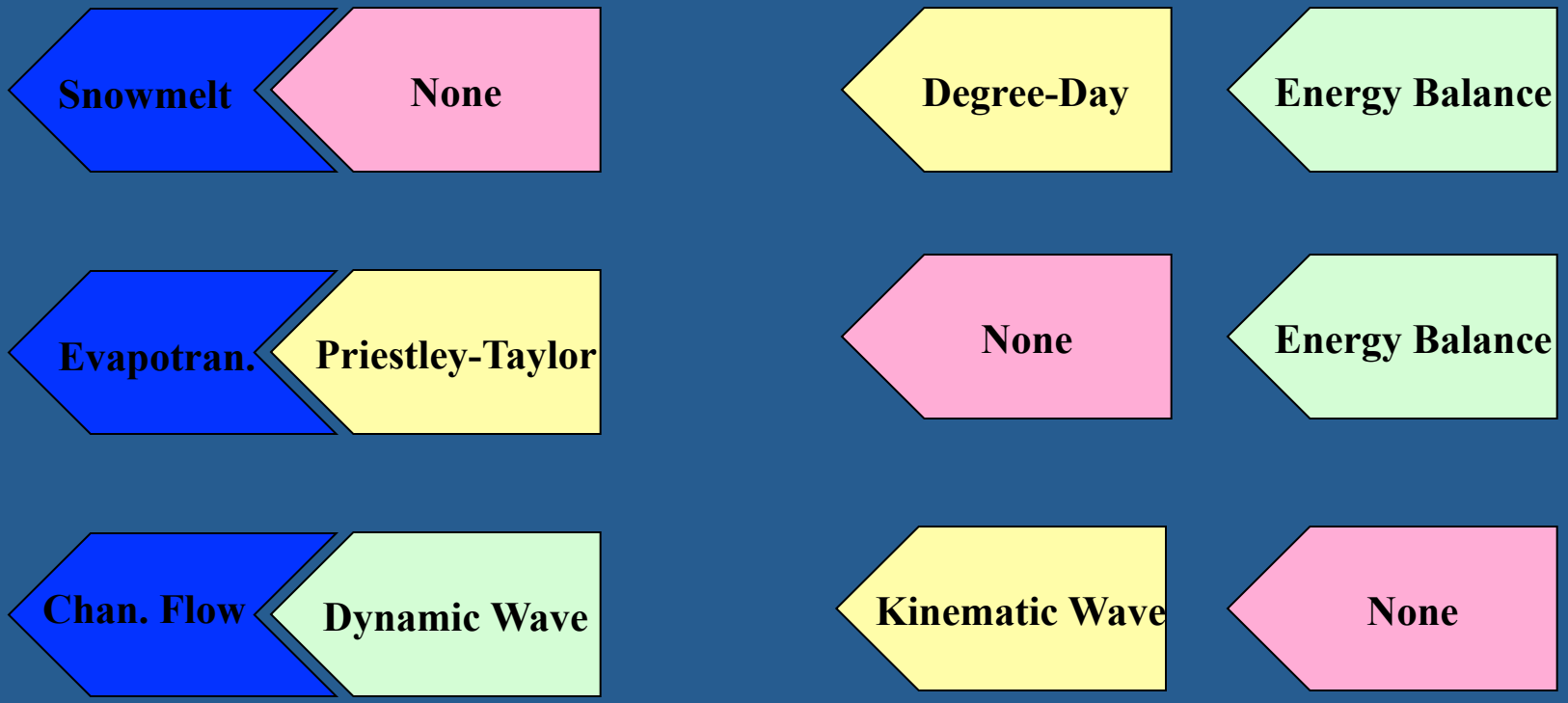
Developer's Overview of the CSDMS Modeling Tool (CMT)

Download it from: http://csdms.colorado.edu

CSDMS Modeling Tool (CMT)



Multiple Methods per Process



Each method has a similar set of dialogs to specify or collect input and output variables. Any process can be turned off.

Infiltration → 1D Richards' Equation (3-layer) Method

The input variables used for modeling infiltration and unsaturated vertical flow with the 1D Richard's equation are defined as follows:

K_s	= saturated hydraulic conductivity [m / s]
K_i	= initial hydraulic conductivity [m / s] (typically much less than K_s)
θ_s	= soil water content at $\psi = 0$ [unitless] (often set to the soil porosity, φ)
θ_i	= initial soil water content [unitless]
θ_r	= residual soil water content [unitless] (must be < θ_i)
ψ_B	= bubbling pressure head [meters] (also called air-entry pressure, ψ_{ae})
ψ_A	= pressure head offset parameter [meters]
λ	= pore-size distribution parameter [unitless] (alt. notation = 1/b)
η	= 2 + (3 * λ) [unitless] (see Notes)
c	= transitional Brooks-Corey curvature parameter [unitless] (see Notes)
dz_{nodes}	= vertical distance between nodes [meters]
n_{nodes}	= number of subsurface vertical nodes

For each variable, you may choose from the droplist of data types. For the "Scalar" data type, enter a numeric value with the units indicated in the dialog. For the other data types, enter a filename. Values in files must also use the indicated units.

Single grids and grid sequences are assumed to be stored as ***RTG*** and ***RTS*** files, respectively. Time series are assumed to be stored as text files, with one value per line. For a time series or grid sequence, the time between values must coincide with the timestep provided.

Equations Used by the 1D Richards' Equation Method

$v = K * (1 - \psi_z)$	= Darcy's Law for vertical flow rate [m / s]
$v_z = J - \theta_t$	= conservation of mass, with source/sink term J
$\Theta_e = (\theta - \theta_r) / (\theta_s - \theta_r)$	= effective saturation or scaled water content [unitless]
$\theta_r = \theta_s (\lvert\psi_B\rvert / 10000)^{\lambda}$	= residual water content [unitless]
$K = K_s * \Theta_e^{\eta/\lambda}$	= hydraulic conductivity [m / s] (see Notes below)
$\psi = \psi_B [\Theta_e^{-c/\lambda} - 1]^{1/c} - \psi_A$	= pressure head [meters] (see Notes below)

Notes on the Equations

1. These equations are used to compute the time evolution of 1D (vertical, subsurface) profiles for (1) soil moisture, θ, (2) pressure head, ψ, (3) hydraulic conductivity, K and (4) vertical flow rate, v. TopoFlow solves these equations separately to get time-evolving profiles for every grid cell in a DEM. The result is a 3D grid for each of these four variables that spans the ***unsaturated zone***. The third equation above just defines a variable that is used in the 4th and 5th equations, so the coupled set constitutes 4 equations to be solved for 4 unknowns. These equations can be combined into one nonlinear, parabolic, second-order PDE (partial differential equation) known as the ***one-dimensional Richards' equation***.

2. The ***infiltration rate*** is simply the vertical flow rate at the ground surface, denoted by v_0.

3. ***Soil moisture*** is simply another term for the water content in the case where the porous medium is a soil.

4. Subscripts in the first two equations indicate partial derivatives with respect to the vertical coordinate, z. Note that z is the vertical distance below the ground surface, in meters.

5. More information on how soil is modeled in TopoFlow along with published soil property tables can be found on this ***soil properties page*** .

Developer's Overview of CMT

- Wireless, automatic linking of components
- Save component configurations and settings as BLD files
  - Uses: save examples, save test cases, collaboration.
  - BLD files can be made to appear in **File > Import Example Configuration**
  - Sample input files on beach at: /data/sims/<model_name>
- Tabbed dialogs and HTML help pages (great resource)
  - Similar user experience for all models
- Launching VisIt remotely
- New driver palette
- Remote file transfer & folder creation
- Launch jobs on the go -- Remotely launch jobs on our HPC cluster (beach) and then disconnect.

Tools to Componentize Models

The main focus of the CSDMS developer team is on making our diverse collection of contributed models reusable as plug-and-play components, in a minimally invasive way. Issues include:

(1) creating a user-friendly modeling environment (CMT)
(2) providing components with a standard calling interface
(3) reading and standardizing input files
(4) creating a "tabbed dialog" graphical user interface (GUI)
(5) acquiring or creating required input files and grids and
(6) saving component output in standard formats (i.e. netCDF)

Design themes include performance, elegance and ease-of-use. We've developed several tools to help us deal with these issues.

Key Design Themes for Plug-and-Play Components

Key Design Themes for Plug-and-Play Components

- Any component can function as **driver** or "**nondriver**"
  - Keeping track of driver/nondriver status
  - Driver must initialize and update all of its "nondrivers"
- Responsible for its own state variables
- Should have its own GUI, input files, help file, etc.
  - Many models use one large input file for all processes
- Avoid unnecessary data transfer between components
  - Imagine that they might communicate over a network
- OpenMI-style status string (give examples)
- Tips for organizing your code
- Component base classes for OO languages: CSDMS_base.py

The Basic "IRF" Interface

In the context of componentized software, **an *interface* is a named set of member functions** that provide a caller with access to its capabilities. (That is, names and data types of all arguments & return values are completely specified.

A basic "IRF interface" is something that virtually all model coupling efforts have in common (e.g. ESMF, OMS and OpenMI). IRF stands for "Initialize, Run_Step, Finalize". **We want contributors to provide this interface.**

```
Run_Model()
    Initialize()
    while not(DONE):  Update()
    Finalize()
```

Initialize() => Open files, read/compute initial values, allocate memory, etc.
Run_step() or Update() => Update all of the computed values (one step forward)
Can call other functions, e.g. Update_Q, Update_v.
Finalize() => Close files, print report, free memory.

Save Model Results in NetCDF Files, Then Use the VisIt Visualization Tool to Make Graphics and Movies

VisIt home page: https://wci.llnl.gov/codes/visit/

Developer's Overview of VisIt

- Open source code under a BSD license
- Client - Server setup with remote **Engine**, local **Viewer**
- Support for 5 dozen formats:
  e.g. Fluent, GDAL (many GIS formats), NetCDF (Basic, Adapt, Lodi, FVCOM), OpenFOAM, VTK, Silo, Tecplot, VTK, etc.
- Improvements to "Basic NetCDF" reader for CSDMS
- Uses multiple processors - Terrascale data sets
- Define new vars with Controls > Expressions (e.g. log(A + 1))
- Complex graphics and easy to make movies
- Python API underneath

Main page: https://wci.llnl.gov/codes/visit/
Summary: https://wci.llnl.gov/codes/visit/about.html
Formats: https://wci.llnl.gov/codes/visit/FAQ.html#12

VisIt as a Python Package

Run "**visit -cli**" to start an interactive Python session and load the visit package.

To visualize a Pseudocolor plot of variable "**var**" from **data_file**, at the Python prompt type:

```
>>> OpenDataBase("<path_to_file>/data_file")
>>> AddPlot("PseudoColor", "var")
>>> DrawPlots()
```

Run "**visit –cli –s <pythonscript_name>**" to load the Python visit package and call the VisIt commands in the script.

In VisIt, choose ***Controls > Command...*** to **Record**, **Write** or **Execute** Python commands that correspond to button clicks.

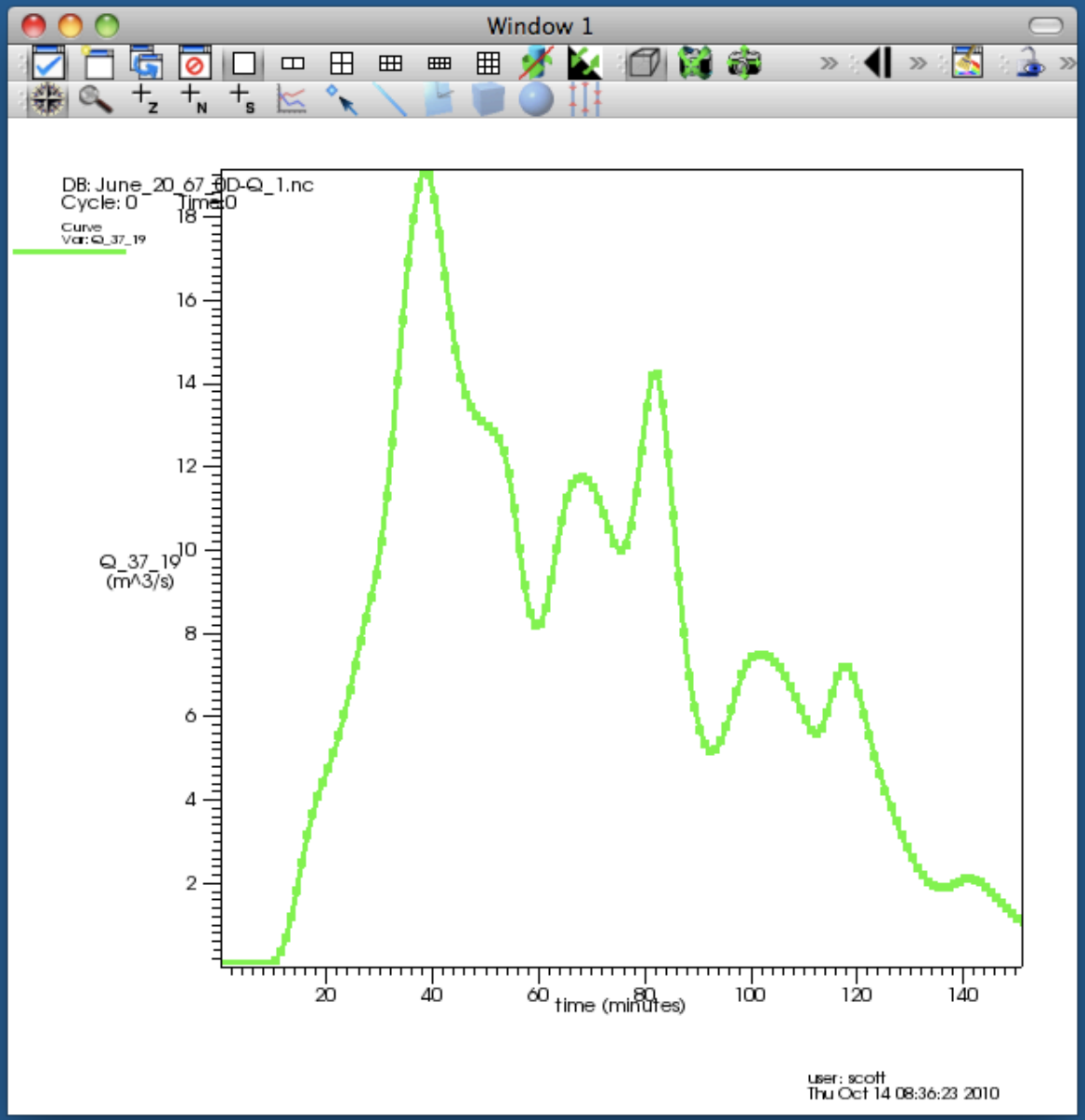
CSDMS Tools for Writing NetCDF

We used the Python package PyNIO to create classes for writing several standard types of output as netCDF files that are CF compliant (mostly).

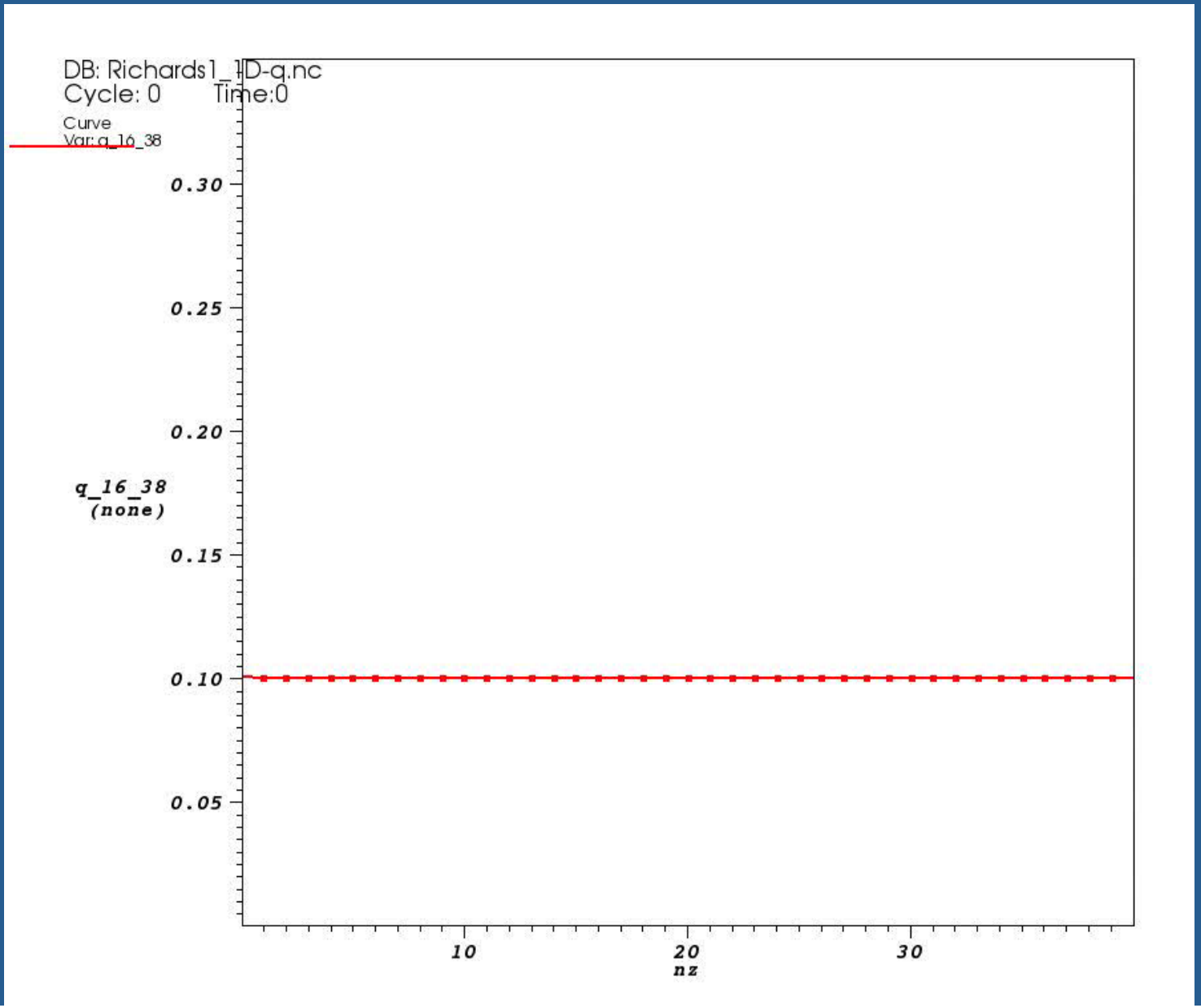
- Time series
- Profile series
- Grid stacks
- Cube stacks
- Scatter plot series (in future)

Regardless of what language your code is written in, you can link to this class via Babel to use the tools.

Writing Time Series to NetCDF

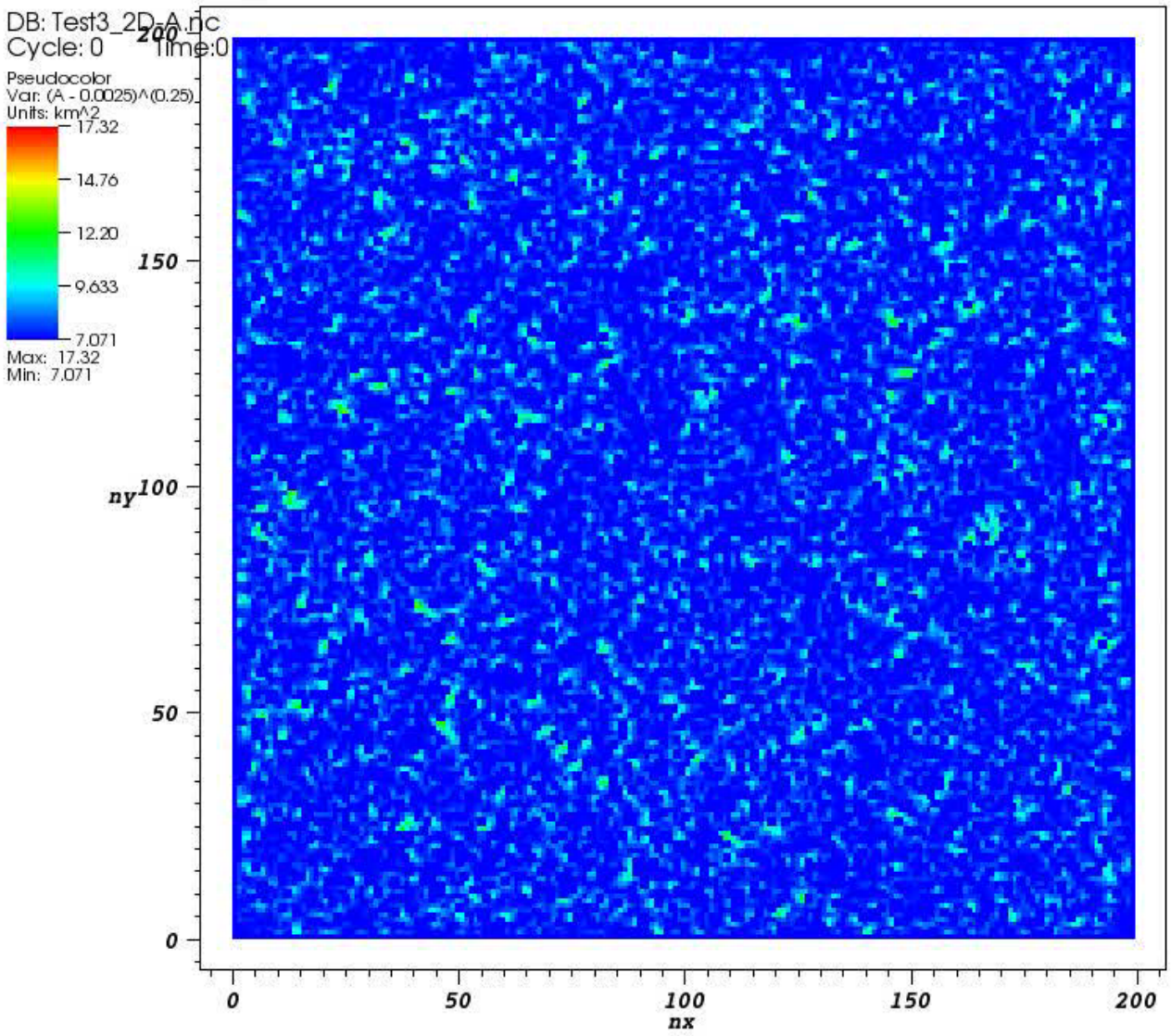


Writing "Profile Stacks" to NetCDF



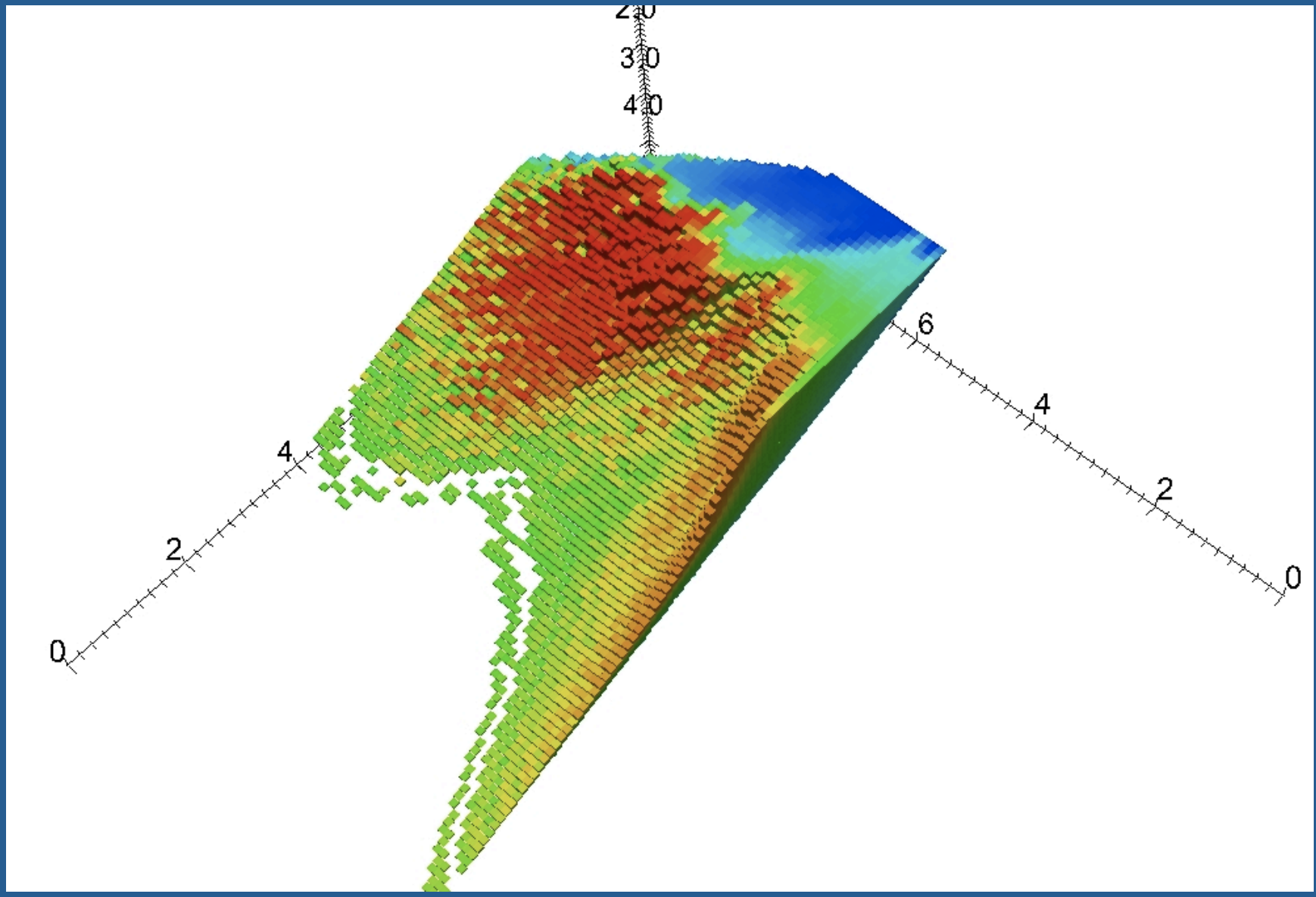
movie

Writing "Grid Stacks" to NetCDF



movie

Writing "Cube Stacks" to NetCDF



Unstructured Grids in NetCDF ?

There is a Google group called the:

UGrid Interoperability Group
http://groups.google.com/group/ugrid-interoperability

that is working on a standardized approach to saving unstructured grid model output as netCDF.

VisIt already supports the FVCOM format, which is a netCDF format.

Talk to Rich Signell, Bert Jaegers for latest news.

Resources Available on Our HPC Cluster "beach"

Resources on Our HPC Cluster

- ganglia – http://csdms.colorado.edu/ganglia/
- transcode – http://www.transcoding.org/
- environment modules – http://modules.sourceforge.net/
- torque –
  http://www.clusterresources.com/products/torque-resource-manager.php
- numerous compilers –
  http://csdms.colorado.edu/wiki/Help:CSDMS_HPCC
- Python 2.6 and many packages –
  (numpy. scipy, matplotlib 1.0, PyNIO, suds, urllib2, visit, and many tools/APIs we've developed)
- PETSc – http://www.mcs.anl.gov/petsc/petsc-as/
- TauDEM – http://hydrology.usu.edu/taudem/taudem5.0/index.html
- VisIt – https://wci.llnl.gov/codes/visit/

New Set of D8-based Tools

Packaged as a Python component/class, with methods for:

- Pit filling (create a depressionless DEM)
- D8 flow direction grids
- Contributing area grids
- Slope grids (pixel-to-pixel), etc.

- d8_global.py and d8_local.py

Already used for TopoFlow (spatial hydro model), Erode (landscape evolution model, DEM Profile Smoother, etc.)

- TauDEM parallel version, too!

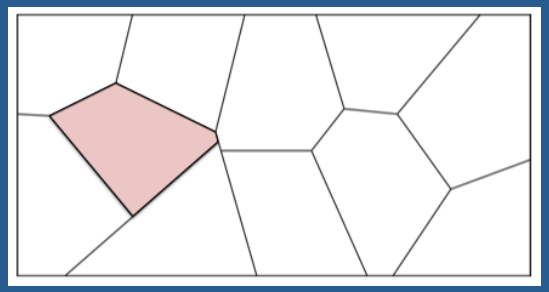
Creating GUIs from XML

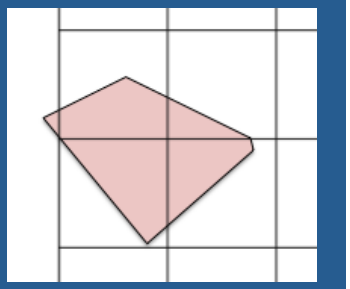
The CSDMS component wrapping process allows a “tabbed dialog GUI” to be created for a component. All that is required is an XML file that describes the tabs, labels, default values, min and max allowed value, etc.

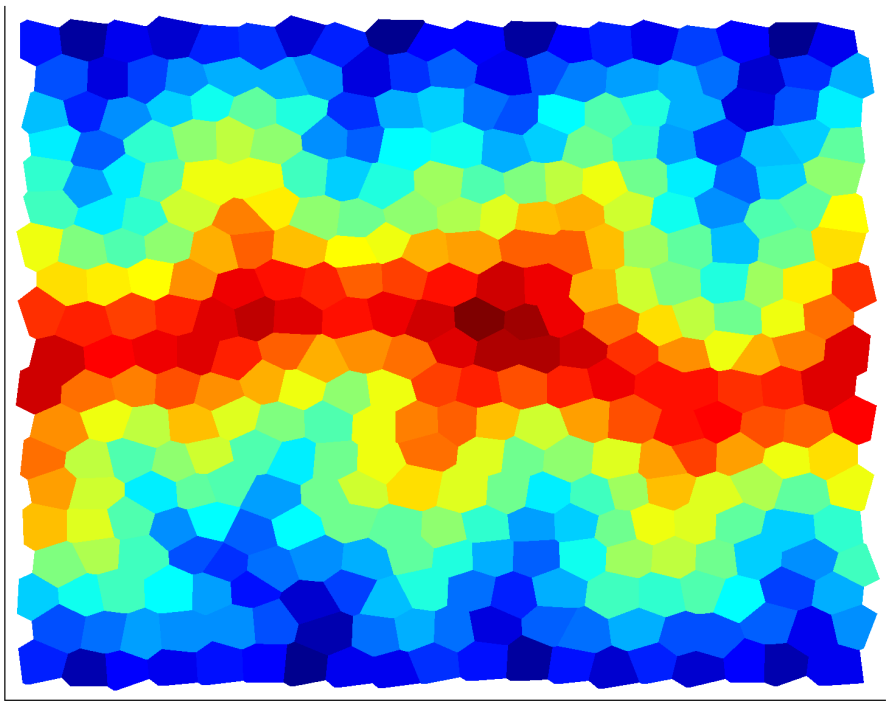
```
<dialog>
<!-- ========================================================================== -->
  <tab name="Project">
    <entry name="/ROMS/Input/Dir">
      <label>Input directory</label>
      <help_brief>Path to input files. {cb;GUI;/data/sims/roms/upwelling}</help_brief>
      <default>GUI</default>
      <type>String</type>
    </entry>

    <entry name="/ROMS/SitePrefix">
      <label>Site prefix:</label>
      <help_brief>Site prefix for input/output files.</help_brief>
      <default>upwelling</default>
      <type>String</type>
    </entry>
```

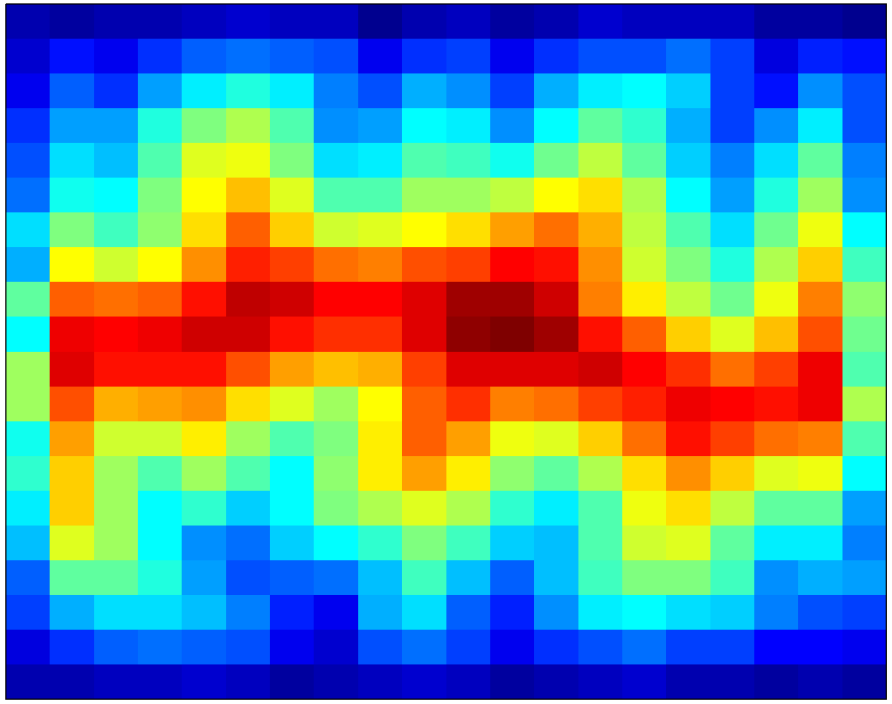
CSDMS Regridding Tools

- ESMF Regrid (multi-proc., Fortran)
- OpenMI Regrid (single-proc., Java)









Python is Really Cool, Too

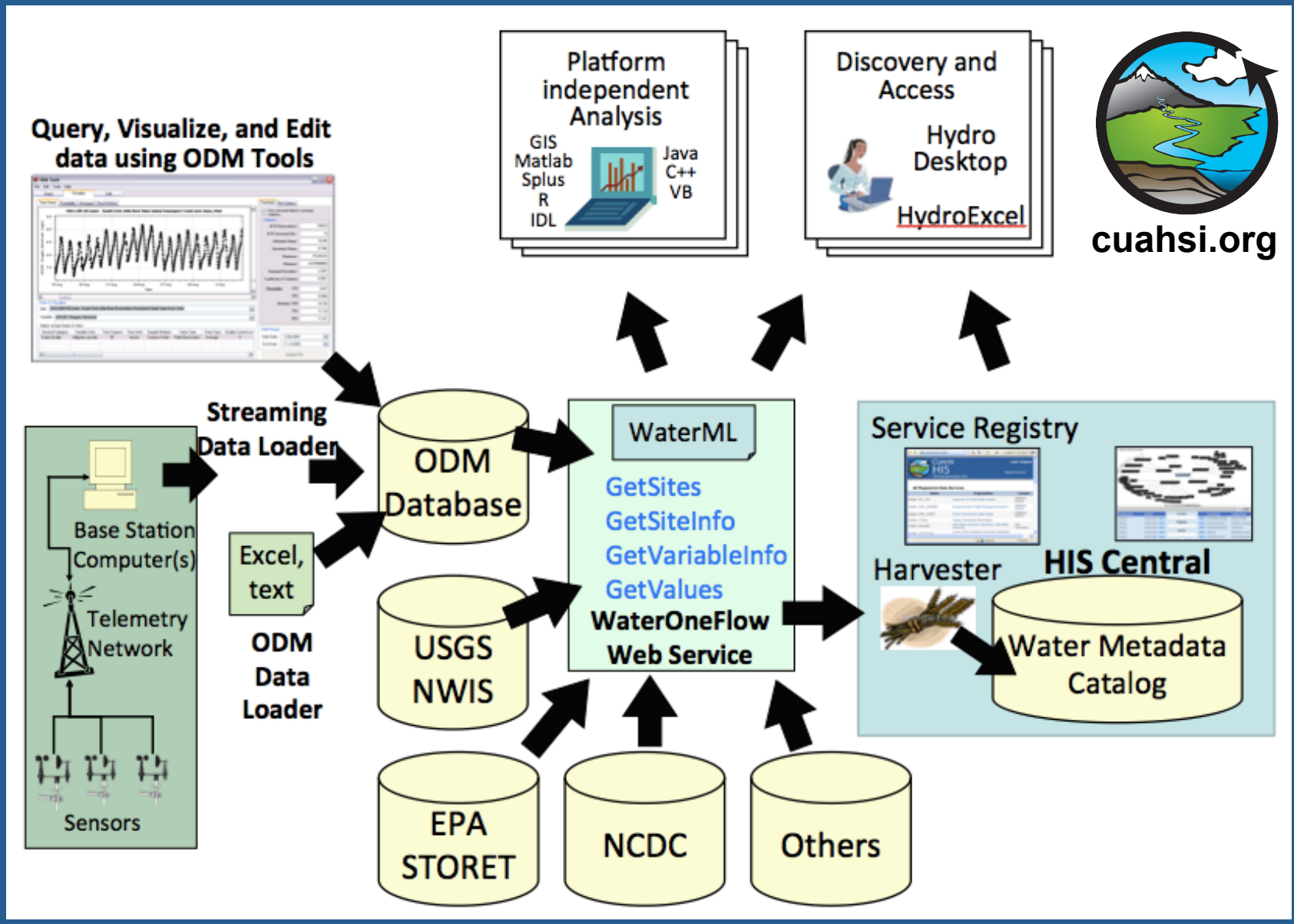
Eastern Brown Snake, Daintree, Australia (Not a python)

The Role of Python in CSDMS

- Open-source work-alike to tools like MatLab and IDL
- Easy-to-learn, but a full object-oriented language
- Great for many types of scripting tasks (i.e. vs. shell scripts)
- Many low-level support tools in CSDMS system
- Is the new scripting language for Arc GIS, etc.
- Can be used as a scripting language for VisIt
- Recent release of matplotlib 1.0 for graphics (very nice)
- Ability to create professional-grade GUIs (wxPython pyQT)
- Provides web-service packages like suds, urllib2, pydap
- Provides an API for netCDF via PyNIO
- Many tools now for going between MatLab and Python
- PyCUDA API for GPU programming
- Adoption by the TelluSim project (LEMs, etc.)

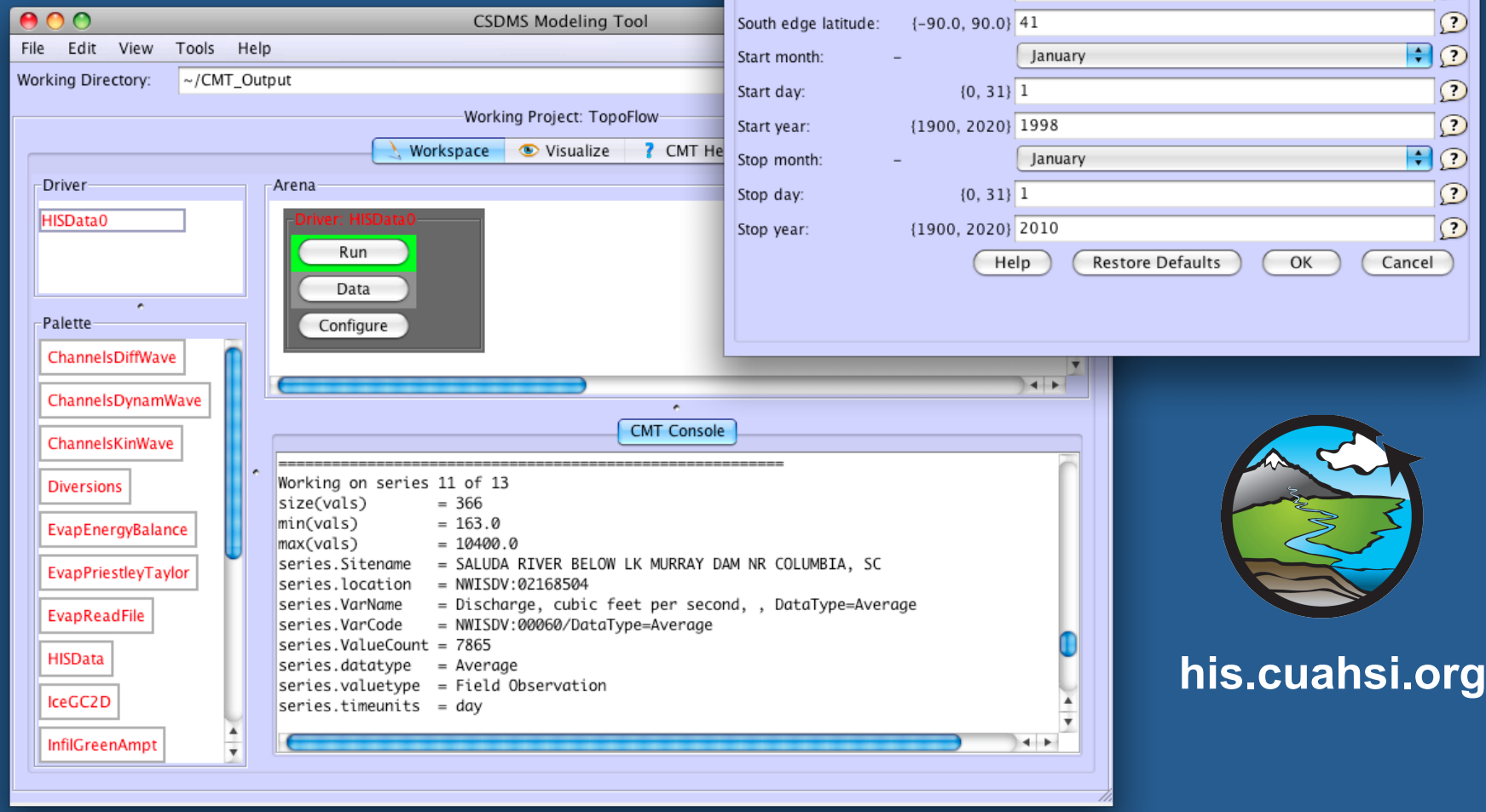
Components That Access Web Services to Get Data

Overview of CUAHSI HIS System



Components that Use Web Services

- New HIS Data component
- Python packages: suds and urllib2



his.cuahsi.org

Future Directions

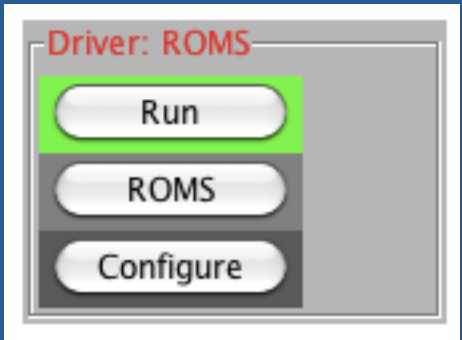
- New hardware on the way:
  - 128 new cores (16 nodes x 2 procs x 4 cores)
  - Status of the 18600+ core system CU is getting

- Plans for a family of components with: ROMS, LTRANS, Grid builder, EcoSim, OpenDAP tool, etc. (with funding from a SURA grant with Carl Friedrichs of Chesapeake Focus Group)

- Ability to save a given model configuration as an executable you can run on your favorite OS. (vs. on our cluster)

- More contributed code turned into components

We have streamlined the process of getting contributed models into the CSDMS system as components.
So a key theme for the near future is:



Einer geht noch
einer geht noch rein
Einer geht noch
einer geht noch rein *

But these also happen to be the lyrics to a German drinking song.

Have a great evening!

* Translation: Another one, another one goes in ...