PyModESt: A Python Framework for Staging of Geo-referenced Data on the Collaborative Climate Community Grid (C3-Grid)

Henning Bergmeyer

German Aerospace Center
Simulation- and Software Technology

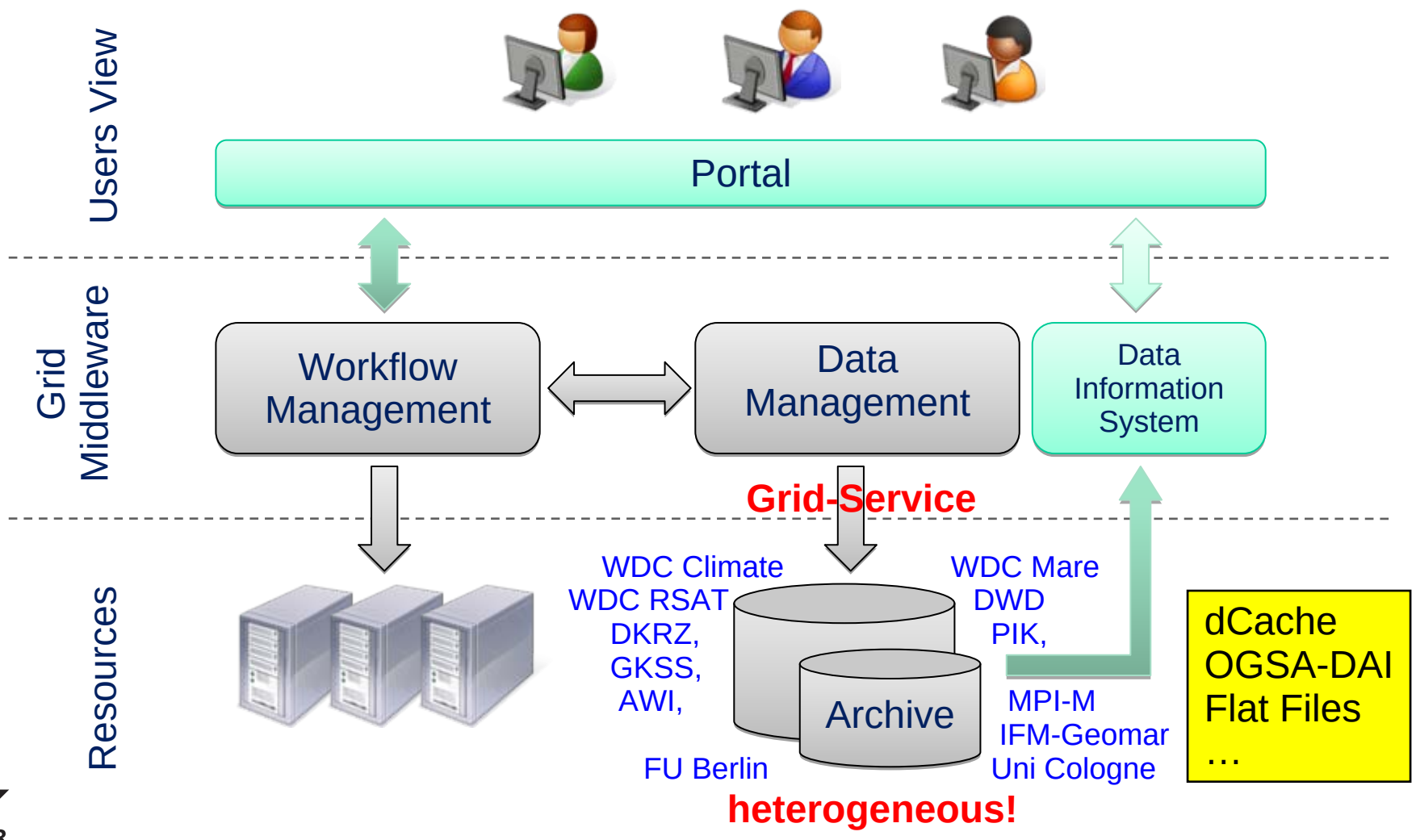
Henning.Bergmeyer [at] dlr.de

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Talk Overview

- *C3-Grid*: A *D-Grid 1* Infrastructure Project
- DLR Use Cases on C3-Grid
- Concept and Realization of Data Providers
- Implementing Staging Scripts with PyModESt
  (**Mod**ular **E**xtendable **St**ager in **Py**thon)
- Interoperability Issues and Goals

C3-Grid Layered Architecture (simplified)

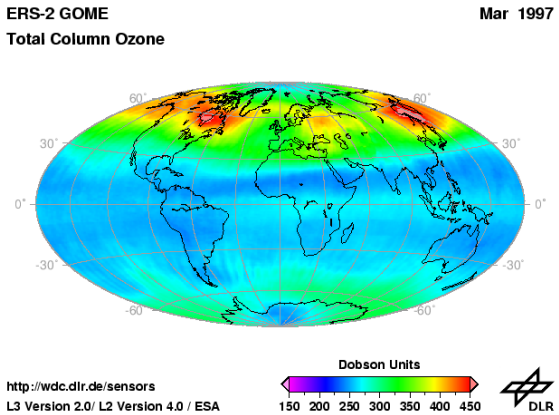


C3-Grid : A *D-Grid 1* Project

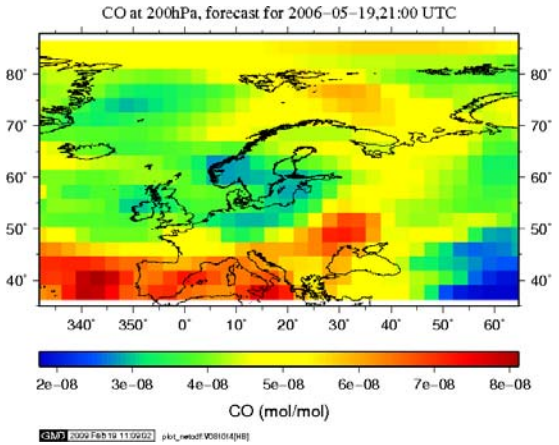
- D-Grid: German Grid Initiative
  - Phase 1: Infrastructures and Services for Scientific Communities
- C3-Grid (2005-2009)
  - **Transparent Grid Infrastructure** for the German Climate Research Community (Globus Toolkit 4.0.8)
  - **Data**: Simplified Uniform *Discovery* and *Retrieval* of Distributed Heterogeneous Data on the Web Portal
  - **Workflows**: Computational Standard Tools for Remote Pre-/Post-Processing and Workflows
  - Partners: Meteorological Data Providers and Users, Computer Scientists
- Some future plans
  - Include more Data- and Compute-Providers
  - Partners contribute to IPCC-AR5 and provide some replicas of the data on C3-Grid
  - Enhance security mechanisms: Shibboleth, SLCs, SAML

DLR Use Cases on C3-Grid

- *World Data Center for Remote Sensing of the Atmosphere*
  - Data Provider for Global Satellite Sensor Data
  - e.g. “Global Ozone Monitoring Experiment“
  - Data Located on Web Server (free)

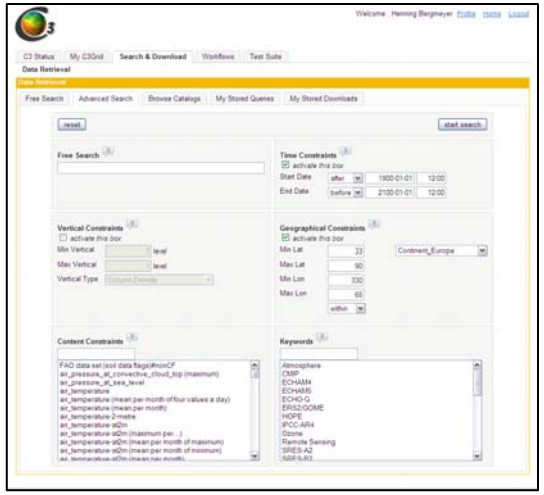


- *Institute for Physics of the Atmosphere*
  - Workflow Developer for Model-Driven Chemical Weather Forecasts for Flight Route Planning
  - Data Provider for Data Selection
  - Visualization Tool
  - Data Located in File System (free)

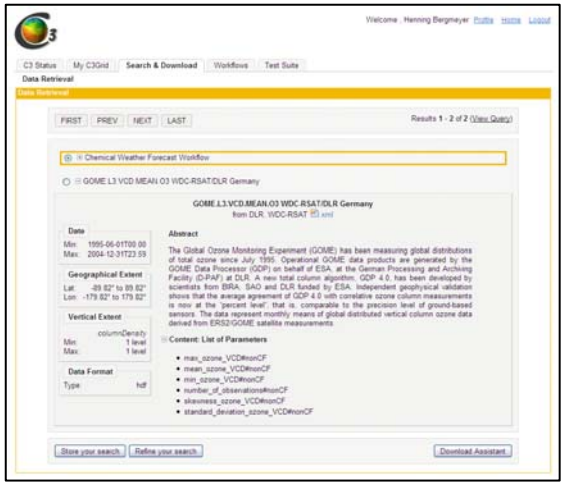


Data Providers I
Portal-Based Discovery

- Advanced Search



- Browse by Data Set

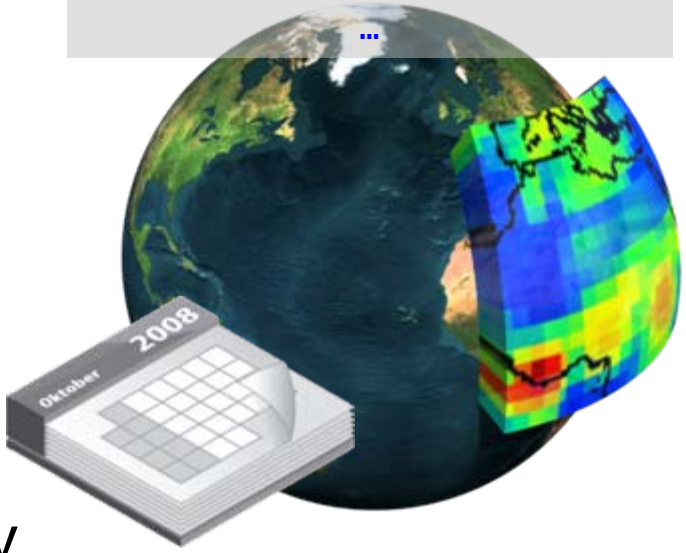


- Available data sets are annotated using I*SO19115/19139 MD Profile*
  - C3-Grid project provides a documented online meta data editor form
- Meta data is published on an *OAI* Server (e.g. DLESE jOAI)
- Meta data is *OAI-PMH* harvested by *Data Information System*
  - Portal Integration: "*panFMP*" (available on *Sourceforge* and *panfmp.org*)

Data Providers II
Uniform Data Access

- Data Download Assistant in Portal
- Grid-Service Receives a Standard Set of Selection Constraints
  - Data Set (*Object ID*)
  - Variables as *CF Names*
    (Climate and Forecast MD Convention)
  - Regional Bounds (Longitude, Latitude)
  - Vertical Bounds and
    Vertical Coordinate Reference System
  - Time Period (not ISO8601 restricted)
  - Data Set Specific Constraints

```
log_surface_pressure
mole_fraction_of_ozone_in_air
…
```



- Data Provider gets transparently
  - *Distinguished Name* of requesting user for authorization mechanism
  - C3-Grid-wide Unique Workflow ID

Data Providers III Delivering Data

- Deliver only the data fulfilling the constraints
  - Extract the corresponding parts of the base data
  - Reduce necessary file size for remote transfers
- Deliver exactly 1 data file and 1 meta data file
  - Always produce meta data for data files
- Provider may offer more than 1 file type (NetCDF, HDF, GRB)
- Compress files on request (.tar.gz)
- Place result in local *DMS work space*
  - GridFTP accessible directory
  - Managed by central C3-Grid DMS

- Estimate time to complete request and needed storage space for result

Becoming a C3-Grid Data Provider

- Prepare Data and ISO Meta Data
- Grid certificate for own server and local grid users
- System Set-up through an Admin
  - Middleware: Globus Toolkit 4.0.8
  - Configure Firewall, authorize Certificate of DMS
  - GNDMS Software of Zuse Institute Berlin
    - Basic installation using *ant*
    - Configuration for local setup using variables in a special shell script
  - MDS-Entry in Resource Information System (RIS)
- Providing Data Sets: Implementation of scripts for Data Staging and Estimation

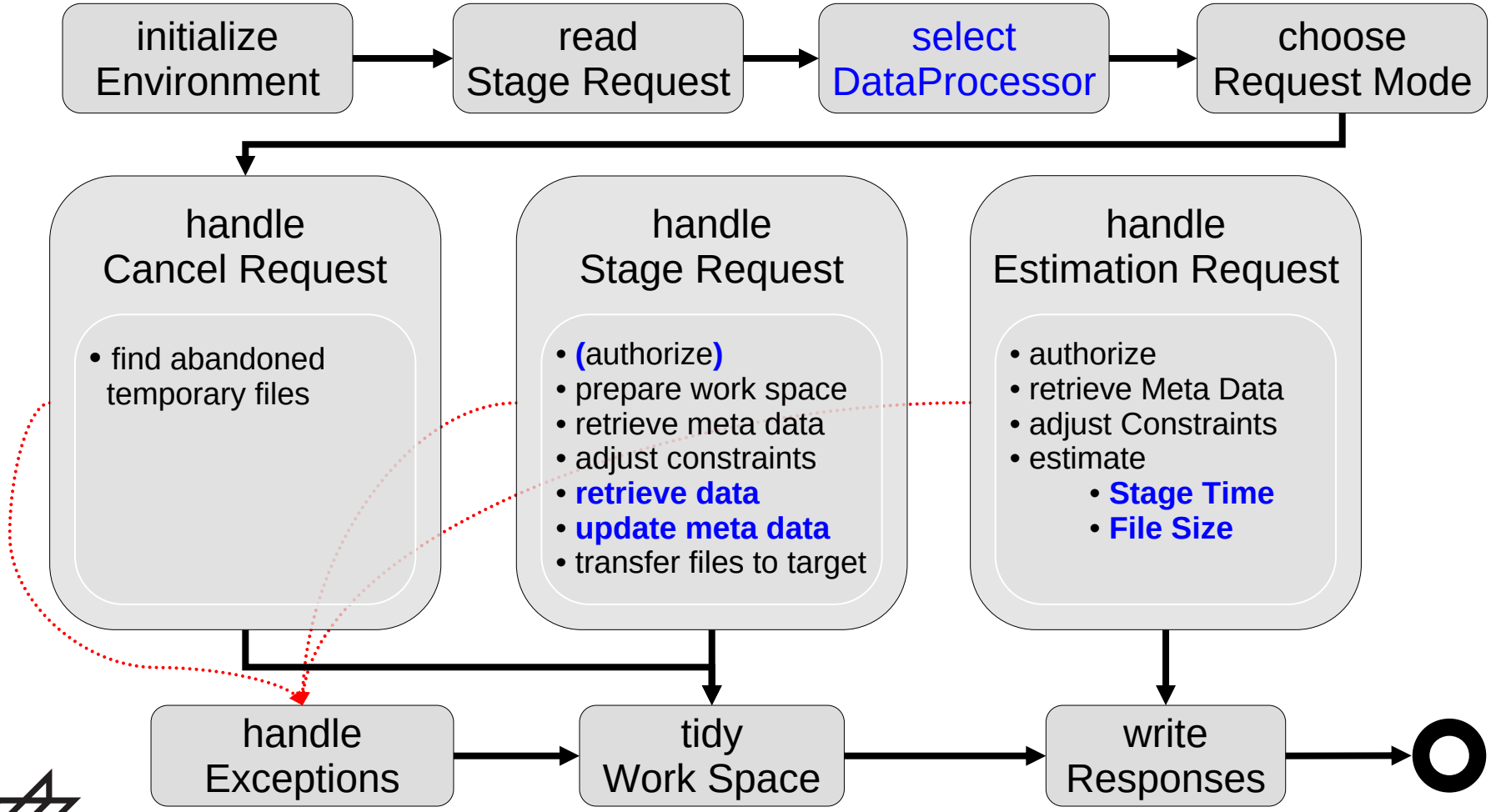
Data Providers know their tools and their data!

General Implementation of File Stagers for C3-Grid

- Receive and interpret request constraints on STD-IN as XML or Property File Format, then fulfill either of
- Case 1: **Estimation Request**
  - Verify constraints, estimate result file size and staging time
  - Offer a contract in Property or XML-Format on STD-OUT
  - Do NOT process any base data files for this
- Case 2: **Stage Request**
  - Retrieve the data and produce one result data file
  - Produce a corresponding meta data file
  - Take care of concurrent service executions when using temporary files
- Case 3: **Cancel Request**
  - Clean-up temporary files from interrupted requests
- Implementation is open and can be done by extending the Grid-Service in Java or by calling any executable as “External Stager”

PyModESt Staging Process Skeleton

Necessary Implementation effort for DP when using PyModESt



What the DP does and What is done for her

- Manually:
  - associate OIDs with data processors
  - associate CF names with variable indices
  - retrieve and package data using your well-known tools
  - enter precise result attributes for meta data update
  - estimate file size and staging time
  - authorize user or deny access
- Automatically:
  - Std-In/-Out communication
  - Stage request validation
  - Complete meta data file handling and operations
  - Creation of python variables for request constraints (float, datetime, str)
  - Temporary file management (preventing concurrency conflicts and storage leaks)
  - Choice of processing method by OID
  - Thesaurus: 2-way variable name translation
  - Compression tar.gz
  - Logging
  - Error handling (log, service response, tidy-up)
  - Gauss grid calculations
  - Calling external tools catching its std-output

PyModESt
Modular Extensibility

- **Data Processor** module:
  - implement data set specific operations
    __init__(c3env, stage_request)
    retrieveAndFilterDataFiles()
    updateMetaData(c3_metadata)
    estimateFileSize() returns long
    estimateStageTime(stage_moment) returns timedelta
  - define variable name associations between data set scopes for helper module *C3Thesaurus*

```
SCOPE_MAP = { "g2.de.dlr.wdc.ERS2.GOME.L3.VCD.MONTHLYMEAN.O3" :
        { "mean_ozone_VCD#nonCF"                : "mean",
          "standard_deviation_ozone_VCD#nonCF" : "strd_dev“ }
}
```

- The **starter script** associates data sets and data processors:

```
PROCESSOR_TYPES = {
   "g2.de.dlr.wdc.CWF"  : netcdf_extraction.IPA_NCDF_Processor,
   “g2.de.dlr.wdc.ERS… “: wdc_hdf_processing.WDCHDFProcessor }
```

PyModESt Metadata Update

```
md.removeQuicklook()

md.filterContentInfo(self.thesaurus.translateToC3(
                     self.requested_vars, self.object_id))

md.setHorizontalBounds(self.lon_min, self.lon_max, self.lat_min,
                       self.lat_max)

md.setVerticalExtent(self.alt_min, self.alt_max,
                     self.alt_verticalcrs)

md.updateTimePeriod(self.timeperiod_begin_date,
                    self.timeperiod_stop_date)

md.setObjectId(self.object_id + "." +
               datetime.utcnow().isoformat().replace(":","-"))

md.addLineageProcessStep(
   PROCESS_DESCRIPTION,
   datetime.datetime.utcnow(),
   self.stage_request.object_ids[0],
   RESPONSIBLE_PERSON,
   "http://wis.wmo.int/2006/catalogues/gmxCodelists.xml#CI_RoleCo
   de_distributor",
   INSTITUTE_IDENTIFIER)
```

Estimation for Offer Contracts

- File Size
  - Use “GaussianGridHelper” to calculate table index ranges on Gaussian Grids

```
gauss_grid_hlp = RegularGaussianGridHelper(
    src_lat_min, src_lat_max, lat_delta, lat_len,
    src_lon_min, src_lon_max, lon_delta, lon_len )

lat_idx_min, lat_idx_max, lat_idx_len,
    lon_idx_min, lon_idx_max, lon_idx_len
    = gauss_grid_hlp.calculateRegionIndices(
          lat_min, lat_max, lon_min, lon_max )
```

  - For Raster / Table Data simply multiply and sum-up
  - Difficult for data on irregular coordinate system (e.g. time series)

- Staging Time:
  - Return a ***datetime.timedelta*** value
  - It is next to impossible to be precise
  - Currently DLR implementations generously over-estimate with a constant: ***timedelta(seconds=60)***

Examples: DLR Use Cases

- WDC-RSAT: ERS2.GOME.L3.VCD.MONTHLYMEAN.O3 (95–05)
  - Base Data: 1 file / month, file format HDF4, HTTP download
  - Retrieval and Processing using ***PyHDF* library**
    - create new HDF file
    - iterate over months covering requested time period
      - adjust data describing attributes

- IPA: Chemical Weather Forecast Demo Data Set (2005)
  - 1 file with 8 time steps / day, file format NetCDF, local file system
  - Retrieval and Processing using external command line tool ***Climate Data Operators***
    - iterate over files corresponding to time period (***cdo***)
    - adjust data describing attributes by analyzing the result file (***cdo***)

Benefits of Using PyModESt

- *Easy to understand* and modify by non computer scientists.
- Intuitive configuration using dictionaries
- No compilation necessary
- Use Python types (*float*, *str*, *dict*, ...)
- Use Python standard libraries (*datetime*, *timedelta*, *math*, …)
- Add new data sets by *Copy and Customize* DataProcessors (documented *DataProcessor* template is provided)
- Easy integration of command line tools (CDO…)
- Rich set of useful libraries available (HDF, Numeric, iso8601, PyParsing…)
- Integration of Java and C/C++ Libraries with JPype and CTypes possible

Open Issues

- CF Names for Variables on C3-Grid level
  - Data sets indices in most cases use institution or file format specific conventions => **automatic translation necessary**
  - Not all used variables have already CF standard names => new names have to be chosen and discussed on community level
    - slows down development
    - difficult to discover when unknown to user
    - requires documentation on portal and reading by user
  - Centralized naming and translation service would be helpful
- Support for better staging time estimation
- Authorization using Short Lived Certificates (Proxies) and SAML assertions (“*GapSLCs”* beginning soon)

Summary

- The C3-Grid is a **collaboration infrastructure** for the climate science community based on the Globus Toolkit 4.0.8 and WSRF grid services.
- Main aim is **transparency of the infrastructure** to users by abstraction of **heterogeneous data resources** for easy data discovery and access and to allow execution of basic manipulation and analysis tasks as well as complex distributed **workflows**.
- ***PyModESt*** is a Python framework for the comfortable **modularized implementation of staging scripts** for the C3-Grid that frees meteorological data providers from doing the work of system admins and vice versa.
- Open issues for interoperability and extensibility are a semantically defined, unified naming and translation service for variable names and (yet) the integration of authorization principles based on Shibboleth, SLC and SAML assertions.