Share Data with OPeNDAP Hyrax: New Features and Improvements

July 2016

James Gallagher (OPeNDAP)

This work is supported by NASA/GSFC under Raytheon Co. contract number NNG15HZ39C and a Grant from the Australian National University.
Overview

- **Aggregation**
  - Performance improvements
  - User-invoked aggregation

- **Authentication**
  - End-to-end (web & programmatic clients)

- **Web-Service Protocols**
  - W10n, WMS
Brief review of Aggregation

• Combine discrete granules (e.g. files)
• They are discrete usually because of the mechanics of collection or processing
• Server aggregation frees users from having to understand the archive’s exact structure
• Data type determines which aggregation techniques are appropriate
  – Regular data – e.g. level 3
  – Swath data – e.g. level 2
DAP Servers offer Virtual Aggregations of **Identically Shaped Variables** in different granules

- Lengthening An existing dimension
- Adding Some New Dimension

*as specified via NCML*
Aggregation: performance

• Test case*: Suppose a user were to ask for all of the data from a small aggregation – 120+ files of AIRX data:
  – The original server would leave the client hanging for ~26 minutes and then drop the connection… (because the network connection times out)
  – The new server will return all of the data (166GB) and begin to stream the results within fractions of seconds!

• What if the hardware were made large enough so that the original software could complete?
  – The original server has an effective transfer rate of 5MB per second…
  – The new server has an effective transfer rate of 33MB per second*, a factor of 6.8 increase!

• Server memory requirements were also reduced
  – The original server used memory (RAM) roughly equal to the response size…
  – The new server has a flat memory consumption that it equal to the smallest atomic unit accessible from the data store (e.g., array slab). For the test case above, the reduction is 4 orders of magnitude!

• Key Points:
  – The server can now handle requests it simply could not before
  – This scales to very large requests – because the implementation uses pipelining to minimize data held in memory at any given time

* Tests run between machines in the Amazon cloud, see notes
Quicker Initial Response

One Variable Selected from Increasing Numbers of Time Slices (Granules)

Baseline
Improved

Elapsed Time (seconds) to Start of Response

Number of Granules (multiple runs for each bar; using the besstandalone test harness)
Less Server-Memory Use

Memory in (log) MB

Test Scenarios

Baseline
Improved

One File, N Vars
One Variable, N Granules
10 Granules, N Vars
Aggregation of swath data
...is also supported, differently

Simple aggregation techniques do not apply to swath data
Aggregation: user-driven

• User-driven: The client provides a list of datasets to aggregate
• Web service interface is easy to customize for differing clients
• While intended for ‘Swath data,’ this can be used with any collection of datasets
• Users may get data from thousands of files in a single operation (e.g., clicking a single link)
• Con: Clients must know the granules they want
User-driven Aggregation Mechanisms

- Two different response forms $\rightarrow$ different aggregation algorithms
  - Iterate over a set of granules applying a constraint to each, collecting discrete results in a single archive file
  - Iterate over a set of granules, transforming them to a table* and selecting a subset of those rows by value

*Actually DAP Sequence objects
User-driven Aggregation Interface Description

• Implemented using a new web service that relies on POST so it can accept larger inputs than HTTP GET will allow.

• It accepts a series of commands that describe the
  – Response format (e.g. files in a zip archive, CSV-encoded table)
  – Datasets, modified by their constraints, to aggregate
User-driven Aggregation Example

• Users of NASA’s Earthdata Search Client* can receive single data responses from queries that involve many datasets.

• Example operations
  – Get the server’s version
  – Get a simple aggregation, returned in an archive, values encoded in netCDF3 (other formats are supported).
  – Get an aggregation with array data transformed to a table, values encoded using CSV

*This feature was initially developed for EDSC
...Server Version

- We can supply arguments using GET
- The simplest ‘operation’ is ‘version’

Request:
http://test.opendap.org/dap/aggregation/?&operation=version

Returns:
Aggregation Interface Version: 1.1
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://xml.opendap.org/ns/bes/1.0#" reqID="[ajp-bio-...">
  <showVersion>
    <Administrator>support@opendap.org</Administrator>
    <library name="bes">3.16.0</library>
    <module name="dap-server/ascii">4.1.5</module>
    <module name="csv_handler">1.1.2</module>
    <library name="libdap">3.16.0</library>
    ...
  </showVersion>
</response>
Returning an Archive

- Use POST; multi-line input

Request:
&operation=netcdf3
&var=Latitude,Longitude,Optical_Depth_Land_And_Ocean
&file=/data/modis/MOD04_L2.A2015021.0020.051.NRT.hdf
&file=/data/modis/MOD04_L2.A2015021.0025.051.NRT.hdf
&file=/data/modis/MOD04_L2.A2015021.0030.051.NRT.hdf

Returns:
Archive: d1.zip
  testing: MOD04_L2.A2015021.0020.051.NRT.hdf.nc  OK
  testing: MOD04_L2.A2015021.0025.051.NRT.hdf.nc  OK
  testing: MOD04_L2.A2015021.0030.051.NRT.hdf.nc  OK
No errors detected in compressed data of d1.zip.
…Returning a Table

- The request looks similar; the return type – CSV – requires different formatting

Request:
```
&operation=csv
&var=Latitude,Longitude,Image_optical_depth_land_and_ocean
&bbox="[49, Latitude, 50][167, Longitude, 170]"
&file=/data/modis/MOD04_L2.A2015021.0020.051.NRT.hdf
&file=/data/modis/MOD04_L2.A2015021.0025.051.NRT.hdf
&file=/data/modis/MOD04_L2.A2015021.0030.051.NRT.hdf
```

Returns:
```
Dataset: function_result_MOD04_L2.A2015021.0020.051.NRT.hdf
table.Latitude, table.Longitude, table.Image_optical_depth_land_and_ocean
49.98, 169.598, -9999
49.9312, 169.82, -9999
49.9878, 169.119, -9999
49.9423, 169.331, -9999
49.8952, 169.548, -9999
49.8464, 169.77, -9999 ...
```
Aggregation – Summary

• ‘Regular’ aggregation performance improved by orders of magnitude
• User-driven aggregation is a new feature
• Regular aggregation – defined by data provider
• User-driven aggregation – defined by user
• User-driven aggregations work on a wide variety of data types
Overview

• Aggregation
  – Performance improvements
  – User-invoked aggregation

• Authentication
  – End-to-end (web & programmatic clients)

• Web-Service Protocols
  – W10n, WMS
Authentication

• Hyrax works with NASA’s Earthdata Login (OAuth2), LDAP and Shibboleth to provide user authentication

• Hyrax + Earthdata Login supports both web and programmatic access
  – EarthData Login access will work for data analysis tools too.
  – LDAP does as well!
  – Shibboleth does not (easily)

• Each of the three are ‘Single Sign On’ services:
  – One database of credentials → many data servers
Authentication - Configuration

• Apache modules provide the actual authentication – Advantage: robust code used by many sites

• Hyrax + Authentication software stack:
  – Apache → Tomcat → Hyrax
  – Configure Apache httpd and Tomcat to work together
  – Configure Apache httpd to authenticate

• Configuration information on the web
  – docs.opendap.org*
Authentication – Programmatic access

• Programmatic clients: put username and password info in a `.netrc` file
• This is better than asking users to install short-lived certificates (a process that they’d have to repeat often)
• This enables automatic access to data
  – Processing done periodically
  – Batch jobs with many accesses
Overview

• Aggregation
  – Performance improvements
  – User-invoked aggregation

• Authentication
  – End-to-end (web & programmatic clients)

• Web-Service Protocols
  – w10n, WMS
Web Services

• Hyrax ships with support for w10n and WMS
• w10n - Webification
  – Use its JSON responses to build/control user interfaces
  – w10n supports navigating collections to get data
  – Its tree model extends into the granules, simplifying UI design & harmonizing data-storage schemes

• WMS
  – Maps: WMS works well with geospatial data that can be shown as a ‘map’ (but not other data types)
  – Google Earth: WMS offers a bridge to other tools…
## Contents of /data/nc

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Modified</th>
<th>Size</th>
<th>DAP Response</th>
<th>Dataset Viewers</th>
</tr>
</thead>
<tbody>
<tr>
<td>bears.nc</td>
<td>2015-10-28T20:08:30</td>
<td>852</td>
<td>ddx dde das das info html rdf</td>
<td>viewers</td>
</tr>
<tr>
<td>bears.nc.das</td>
<td>2015-10-28T20:08:30</td>
<td>145</td>
<td>- - - - - - - -</td>
<td>-</td>
</tr>
<tr>
<td>coads_climatology.nc</td>
<td>2015-10-28T20:08:30</td>
<td>3114044</td>
<td>ddx dde das das info html rdf</td>
<td>viewers</td>
</tr>
<tr>
<td>feb.nc</td>
<td>2015-10-28T20:08:39</td>
<td>5992</td>
<td>ddx dde das das info html rdf</td>
<td>viewers</td>
</tr>
<tr>
<td>fnoc1.das</td>
<td>2015-10-28T20:08:30</td>
<td>162</td>
<td>- - - - - - - -</td>
<td>-</td>
</tr>
<tr>
<td>fnoc1.nc</td>
<td>2015-10-28T20:08:30</td>
<td>23944</td>
<td>ddx dde das das info html rdf</td>
<td>viewers</td>
</tr>
<tr>
<td>fnoc1.nc.html</td>
<td>2015-10-28T20:08:30</td>
<td>177</td>
<td>- - - - - - - -</td>
<td>-</td>
</tr>
<tr>
<td>jan.nc</td>
<td>2015-10-28T20:08:39</td>
<td>6580</td>
<td>ddx dde das das info html rdf</td>
<td>viewers</td>
</tr>
</tbody>
</table>

THREDDS Catalog XML

Hyrax development sponsored by NSF, NASA, and NOAA

**OPeNDAP Hyrax (1.12.2)**

**Documentation**
Dataset: coads_climatology.nc
(http://test.opendap.org:80/opendap/hyrax/data/nc/coads_climatology.nc)

Java Web Start Applications
- Integrated Data Viewer
- NetCDF Tools User Interface

Web Services
- DAP2 Service
- DAP4 Service
- Godiva WMS GUI
  - Web Mapping Service
  - w10n Service

Hyrax development sponsored by NSF, NASA, and NOAA

OPeNDAP Hyrax (1.12.2)
Documentation
Variables

**COADSX [COADSX=0..179]** (Type is Float64)
- attributes

**COADSY [COADSY=0..89]** (Type is Float64)
- attributes

**TIME [TIME=0..11]** (Type is Float64)
- attributes

**SST** (Type is node)
- no attributes
- members

**AIRT** (Type is node)
- no attributes
- members

**UWND** (Type is node)
- no attributes
- members

**VWND** (Type is node)
- no attributes
- members
{
  "name": "COADSX",
  "type": "Float64",
  "attributes": [
    {
      "name": "units",
      "value": ["degrees_east"]
    },
    {
      "name": "modulo",
      "value": [""
    },
    {
      "name": "point_spacing",
      "value": ["even"],
    }
  ],
  "w10n": [{"name": "spec", "value": "draft-20091228"}, {"name": "application", "value": "Hyrax-1.12.2"}, {"name": "type", "value": "dap.2"}, {"name": "path", "value": "/opendap/w10n/data/nc/coads_climatology.nc"}, {"name": "identifier", "value": "COADSX"}],
  "output": ["json"],
  "application/"],
  "name": "dods",
  "value": "application/octet-stream",
  "name": "nc",
  "value": "application/x-netcdf",
  "name": "nc4",
  "value": "application/x-netcdf;ver\u003d4"]
}
Dataset: coads_climatology.nc

Java Web Start Applications
  - Integrated Data Viewer
  - NetCDF Tools User Interface

Web Services
  - DAP2 Service
  - Godiva WMS GUI
  - w1On Service

Hyrax development sponsored by NSF, NASA, and NOAA
SEA SURFACE TEMPERATURE

Layer: My ncWMS server > lds/data/coads_climatology.nc > SEA SURFACE TEMPERATURE
Units: Deg C

Date/time: 16 Dec 0000 01:20:06 UTC first frame last frame

Select date

Fit layer to window

Open in Google Earth
This work is supported by NASA/GSFC under Raytheon Co. contract number NNG15HZ39C