OPeNDAP
Roadmap

Charting New Server-Side Capabilities and Other Supports for Data-Intensive Science
by
Dave Fulker and James Gallagher

OPeNDAP Is:

✴ An acronym = Open-source Project for a Network Data Access Protocol
✴ A not-for-profit corporation that:
  ★ Develops specs for a widely-used, Web-based data-access protocol (DAP)
  ★ Develops/supports server & client software realizing this & related protocols
OPeNDAP Origins

- 1993 – Univ of RI & MIT began work on the Distributed Oceanographic Data System (DODS)
- Data Access Protocol (DAP v. 2) was an outcome
- Results also included clients (the netCDF “client library,” Matlab & IDL) + the first DODS server
- A small developer community emerged
  - Some used DODS toolkits to build clients/servers
  - Others built toolkits (PyDAP, in Python, e.g.)

OPeNDAP, Inc.

- OPeNDAP was incorporated in 2000 as a tax-exempt not-for-profit
- This separated protocol development from domain-specific efforts (at U of RI, & elsewhere, e.g.)
- The OPeNDAP name now completely replaces DODS & may refer to
  - The company
  - The software
  - The protocol, as in “PyDAP is an OPeNDAP server”
Why Web-based Data Access?

✴ Acquiring large datasets may be unwise or highly impractical (too much data)
✴ Web access to subsets is a great solution if
  ★ Client & server agree on the form(s) of subsetting
  ★ The form(s) meet users’ needs (which, in general, rules out subsetting by file selection)
✴ Web-based “publication” might reduce the problem of dark data (from “small” science)

OPeNDAP’s Protocol

✴ DAP2 combines a data model & operators
  ★ The data model includes atomic types (integers, reals, strings, etc), arrays, structures, grids & sequences
    ✦ All but the last two are akin to datatypes in C or Java
    ✦ Grids & sequences are relational types
  ★ Operators offer means to subset all but atomic types
✴ These computer-language-like semantics
  ★ Simplify client programming
  ★ Yield discipline neutrality
Typical Applications

✴ DAP2 is generally implemented as a web service, with requests made by constructing a URL
✴ DAP2 responses are “documents” of two types
★ Text-only descriptions of datasets or catalogs (metadata only)
★ Combinations of (textual) metadata & binary data values
✴ Clients use these responses in various ways, e.g.:
★ Matlab incorporates the documents into its workspace, assigning them its own internal data types
★ The netCDF client library makes (legacy) applications believe they are reading from a local file

Operations:
Selection & Projection

✴ DAP2 “constraint expressions” invoke selection and/or projection operations
★ Both are optional; omission yields entire dataset
★ Projection defines subsets by their indices
★ Selection defines which (named) variables are returned by their values
✴ Applied to grids or arrays, projections may alter their structure (dimensionality, e.g.)
Selection constraints can be used to return subsets defined by (ranges or decimations of) index values.

The Client-Side View

- OPeNDAP clients acquire data via URLs
  - Retrievals are self describing (i.e., they are metadata-rich)
  - Catalog views often simplify data discovery
- Well-designed clients (MATLAB & IDL, e.g.) retrieve only subsets, per their users’ actions
- Very fast for lower-dimension subsets
  - 2-D fields from a 4-D simulation, e.g.
Server-Side View

The OPeNDAP server (called Hyrax):

- Robust; secure; efficient; easy to install; and well supported
- Realizes a “standard” protocol (per NASA, others)
- Offers a unified view of multi-formatted data!

Remote/distributed holdings facilitate:

- Delegation of data-curation/preservation roles
- Tracking of usage & data-flaw discoveries

and there’s more...

“Aggregation”

Hyrax can aggregate many files into a logical dataset that clients access via one URL

- Much preferred over thousands of filenames!
- Reasonably efficient subsetting in any dim

Same mechanism can wrap old datasets with improved or augmented metadata
Planned New Features

✴ Better support for polygonal meshes
  ★ Subsetting is a major challenge...
✴ Asynchronous access mode
  ★ Delayed retrievals of, e.g., “near-line” data
✴ Specification-conformance testing
  ★ Also applicable to non-OPeNDAP servers using DAP
✴ Access policies that vary by user

Challenges of Subsetting Polygonal-Meshes (UGRIDs)

✴ Polygonal meshes (“unstructured grids” or UGRIDs) are not among the DAP datatypes
✴ We must choose a specific way or set of ways (agreed by client & server) to represent these, using the supported data types
✴ Subsetting must be done server-side because
  ★ Datasets are too large to download
  ★ The subsetting operations are quite complex, in general
A UGRID Illustration

- Subsetting should be defined by polygonal regions
- The operation must yield another unstructured grid
- The subset must preserve the topological & geometric relationships
- I.e., we can’t just regrid all to a convenient form

Contemplated Features

- Aggregation of more complex collections
- Inventories (more detailed than catalogs)
- Polygonal-mesh subsetting \(\rightarrow\) (?) other extended server-side query responses
  - Linking to semantic services, predicate logic...
  - Re-gridding & other computations (specified how?)
  - Support for social aspects of data exchange/use/reuse