OPeNDAP

Current & Planned Client-Server Capabilities for Data-Intensive, Transdomain Science
by
Dave Fulker, President, OPeNDAP, Inc.
for
eResearch 2011

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) that dates to ’93
  - Develops/supports client and server software implementing this & related protocols
OPeNDAP Origins

- 1993 — Univ of Rhode Island & MIT launched the Distributed Oceanographic Data System (DODS)
- Early results included
  - Data Access Protocol + 1st DAP-compliant server
  - Clients: MATLAB, IDL & the netCDF “client library”
- A small developer community emerged
  - Some used DODS toolkits to build clients/servers
  - Others built toolkits such as PyDAP (in Python)

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 forms of subsetting
  - The forms 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 Data Model

- Variables have names, attributes & values:
  - Atoms (integers, reals, strings, etc) are used to build (Java- or C-like) structures & arrays
  - These combine into sequences & (n-dim) grids
- Operators—selection & projection—can yield subsets of all but the atomic data types
- Though broadly sufficient, these basic semantics simplify programming & exhibit discipline neutrality

Web Services on DAP

- Requests made by URLs (with constraints)
- Responses are “documents” of two types
  - Textual descriptions of data/catalogs (metadata only)
  - Combinations of (textual) metadata & binary values
- Clients use these responses in various ways, e.g.:
  - MATLAB maps documents onto its internal math types
  - The netCDF client library allows (legacy) applications to act as though reading from a local file
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

Remote/distributed holdings facilitate:
- Delegation of data-curation/preservation roles
- Tracking of usage & data-flaw discoveries

The OPeNDAP server (called Hyrax):
- Is robust; secure; efficient; easy to install; well supported
- Realizes a “standard” protocol (per NASA, others)
- Offers a unified view of multi-formatted data!

Transdomain Brokering

Hyrax servers broker transactions between domains with dissimilar data “currencies”

Practical because the DAP data model is
- Rich enough to serve as a common currency
- Simple enough to enable building/fitting of
  - Client-side tools for analysis, visualization…
  - Server-side (back-end) adaptors for legacy data
- Reduces brokering problem to order $N$ (from $N \times N$)
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 dimension*
- The same mechanism can wrap a legacy dataset with improved/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 (AKA unstructured grids or UGRIDs) are not among the DAP’s data types
- Need means—agreed by client & server—to represent these with supported data types
- Subsetting must be done server-side
  - Subsetting operations are, in general, complex
  - Datasets are too large to download

A UGRID Illustration

- Subsetting should be defined by polygonal regions
- The operation must yield another unstructured grid
- Subsets must retain geometric & topological 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?)
  - Supporting social aspects of data exchange, use & reuse

I thank you.