RFC: Support OPeNDAP in HDFView

03/22/2013

Peter Cao

The HDF Group

# Introduction

One of the requests from 2012 NASA briefing is to use HDFView as a client to access files at an OPeNDAP server. As mentioned at the briefing, we may have more and more requests for this feature. This RFC is to identify what need to be done and how it can be done in order to allow HDFView access files at an OPeNDAP server.

## More answers needed

Before we jump to any implementation, we will need more clarification from our NASA users <need help from Joe Lee or Kent Yang>:

* If any browser can be used as an OPeNDAP client, why do we need HDFView to support OPeNDAP (or is this work important)? <Kent’s comments: *This is a very good question. Given that there are many widely used OPeNDAP clients, what’s the advantage of having HDFView to access OPeNDAP? HDFView, in its name, is a viewer or editor for HDF? Why do we need it to access OPeNDAP? So I think, before the support of OPeNDAP in HDFView, we should have a note and ask typical NASA stakeholders why this work is important. Personally I don’t think this work is important right now. However, I may be wrong*.>
* How should the feature be deployed (part of HDFView release or separate plugin) and maintained (long-term support)?

## Brief review of the OPeNDAP model

OPeNDAP data model includes three basic components: DDX, DataDDS, and Client APIs, as shown in the figure below.

DDX

DataDDS

OPeNDAP Server

Data Files

Client API

Client

DDX -- the XML representation of the OPeNDAP Data Descriptor Object (DDS). It combines the information of the old-style Data Description Structure (DDS) and Data Attribute Structure (DAS) in a single XML file. DDX is the "ancillary data" that explains what the data is. For example, information of data type, data space, and attributes.

DataDDS -- Data Dataset Descriptor Structure (DataDDS) holds data values along with name and datatype information. The client receives the actual data values (data stream) by DataDDS not by DDX.

Client API – the OPeNDAP project provides a set of client API functions for developing DAP client applications. The current implementation includes OPeNDAP C++ API and OPeNDAP Java API.

To support OPeNDAP in HDFView, we will need to implement a data access layer that connects HDFView and OPeNDAP.

## Requirements

<Kent’s comments: *Information review should be pending before we know why we need to support OPeNDAP in HDFView*>

We will start a couple of the basic requirements. We will need talk to NASA users to get a complete set of requirements.

* Files at the server can be accessed with a simple URL from HDFView. No additional setting is needed.
* Content of remote files will be presented the same way as local files. The way of how data is retrieved from a remote server is completely transparent to users.

## Limitations

* Only HDF4/5 files will be supported at the first stage
* Users cannot write/update the content of remote files. Only read-access is allowed.

# Proposed approach

OPeNDAP project provides Java-DAP2: a pure Java implementation of the DAP2 Protocol. What we need is an HDF-Java object layer based the OPeNDAP Java client API so that data from server can be displayed in HDFView the same way as local files. The following figure illustrates the connections between HDFView and OPeNDAP.

HDFView

DataDDS

DDX

OPeNDAP Server

OPeNDAP Java API

HDF-Java DAP Objects

Data Files

The critical part is to build the layer of HDF-Java DAP objects. There are three basic data objects needed: DapFile (extends FileFormat), DapGroup (extends Group), and DapDataset (extends Dataset). DapFile holds the file metadata and file structure. DapGroup is used to organize the hierarchical structure of a file. DapDataset contains data values and information about the data, such as data type and data space.

Metadata objects are the additional objects that hold the information about the basic data objects. The metadata objects include DapAttribute, DapDatatype, and DapDataspace. The structure of an DapAttribute is very similar to DapDataset, but it contains user-defined metadata for groups and datasets. DapDatatype describes the data type of the dataset, such as data type class, order, and size. DapDataspace specifies the sizes of the dimensions of dataset.

The client/server message/data flow will follow the steps below:

1. A user enters an URL pointing to a file at the server
2. A client file object, DapFile, will be created and encoded to a DAP message using the DAP Java API
3. The DAP message is sent to the server and the server will return the file information as DDX
4. The DapFile object will decode the DDX information and fill the content of DapFile. Objects of DapGroup and DapDataset will be created along with attributes
5. The user now can browse through the file (DapFile object) just like a local file.
6. When the user choose to see the values of a dataset, the data content will be transferred from the server to the client object, DapDataset, by DAP data stream.

# Tasks

The work will include the following tasks. We will need further study to have a good estimation for work of these tasks.

* (20 hours) Study and learn the OPeNDAP model, especially the Java client APIs, and figure how to map DAP message and data to HDF-Java objects. We will need a technical RFC.
* (80 hours) Implement HDF-Java layer for OPeNDAP
  + DapFile
  + DapGroup
  + DapScalarDS
  + DapCompoundDS
  + DapDatatype
  + …
* (40 hours) Integrate the DAP objects in HDFView
* (20 hours) Run testing on various servers and files
* (20 hours) Write users guide and other related documents

# References

[1] <http://earthdata.nasa.gov/sites/default/files/field/document/ESE-RFC-004v1.1.pdf>