4B.4

ADVENTURES IN WEB SERVICES FOR LARGE GEOPHYSICAL DATASETS

Joe Sirott
Sirott and Associates, Seattle, WA

1. Introduction

The amount of data in geophysical data archives is increasing rapidly. For example, the data archive from climate model simulations used for the IPCC AR4 report consumes about 35 TB of storage and it is anticipated that the next round of simulations will require at least a petabyte of storage. It is a major challenge to provide "user friendly" Web access to these data archives.

Dapper (http://dapper.pmel.noaa.gov/dapper/) is a Web server that provides OPeNDAP access to both gridded and in-situ geophysical datasets. It includes DChart (http://dapper.pmel.noaa.gov/dchart/), a Web interface that allows users to visualize and download these datasets. We describe some of the features of Dapper that make it suitable for serving large datasets.

2. Aggregation

Many large datasets contain thousands or even millions of individual data files. Most users are only interested in accessing a small subset of these files. Traditional OPeNDAP servers present all of these files to clients, making it difficult for users to find the variables they are interested in.

Dapper supports two kinds of aggregations: *variable* (also know as a *union* in the Thredds Data Server [1]), and *outer* (referred to as *joinExisting* in TDS). The first allows variables from different files to be combined into one logical dataset, while the second concatenates variables along a coordinate variable (typically time). Outer aggregations can be nested in variable aggregations. Dapper can aggregate local files, or datasets that are available on the Web via OPeNDAP or HTTP.

We have successfully aggregated[2] ~80GB and 3700 files of NCEP/NCAR daily reanalysis data [3] into 5 virtual datasets and ~2TB and 2500 files of monthly mean climate data from the IPCC AR4 model archive at GFDL[4] into 100 datasets.

3. Data streaming

Aggregation can result in virtual datasets that are much larger than the source datasets. A client OPeNDAP request for a variable can require the server to deliver several GB (or more) of data. Many currently available OPeNDAP servers are unable to deliver such large chunks of data to clients in one request because they read all of the data into memory before delivering it to the client. If the request is larger than the amount of memory available to the server or there are many requests occurring at one time the request can fail non-deterministically. This behavior also increases the latency of OPeNDAP requests.

Dapper, by contrast, streams the data to the client. As a consequence, the maximum amount of data that Dapper can deliver is only limited by the maximum size that is supported in the OPeNDAP protocol. The same streaming machinery is also used in DChart to allow users to download large data subsets as netCDF files.

4. HTTP access

Although Dapper can aggregate remote datasets using the OPeNDAP protocol, many data centers only provide HTTP access to their archives. In fact, IT security constraints have forced some data centers to entirely eliminate OPeNDAP access. Dapper can still aggregate data from these archives by using HTTP. We have also added a proxy service to Dapper that translates OPeNDAP requests into HTTP requests and translates HTTP responses into OPeNDAP responses. An OPeNDAP client only has to append the URL of a remote dataset to a Dapper URL to retrieve the data. For instance, a server with a file at http://www.example.com/data.nc could access the data via OPeNDAP by using the URL http://dapper.pmel.noaa.gov/dapper/http://www.example.com/data.nc.

HTTP access to remote datasets is implemented in Dapper with the Unidata NetCDF Java library[5]. We found that we had to make a number of performance enhancements to this library in order to aggregate large datasets. Network latency was a problem, so we created a multithreaded producer/consumer queue to deliver data to the client while simultaneously making HTTP requests to the data archive. Frequently requested coordinate and metadata is cached when the server is started rather than waiting for a client request.
Finally, the Unidata library uses a page-based strategy for HTTP access. This can be very inefficient when accessing time series in large netCDF datasets. We found that batched, variable size HTTP range requests were more efficient.

5. References

[1]http://www.unidata.ucar.edu/projects/THREDDS/

[2]http://dapper.pmel.noaa.gov/dapper/aggregations/www.cdc.noaa.gov/Datasets/ncep.reanalysis.dailyavgs/

[3]http://www.cdc.noaa.gov/cdc/reanalysis/

**Corresponding author address:* Joe Sirott,
7600 Sand Point Wy NE, Seattle, WA 98115;
email: Joe.Sirott@noaa.gov

[4]http://data1.gfdl.noaa.gov/

[5]http://www.unidata.ucar.edu/software/netcdf-java/