

CEOS WGISS: Lessons Learned From WCS Server Design And Implementation

Wenli Yang, George Mason University

Min Min, George Mason University

Daniel Holloway, OPeNDAP.org

Yonsook Enloe, SGT, Inc.

Christopher Lynnes, NASA/GSFC

DRAFT ! Sept 23, 2008

Introduction

The following documents lessons regarding WCS server implementation from a NASA ACCESS project to provide a gateway between the OPeNDAP and WCS protocols. In a nutshell, the project developed a data handler for the OPeNDAP server that enabled serving of data obtained from a WCS server. The project was user driven, by the Coordinated Enhanced Observing Period (CEOP) community, which also afforded us the opportunity to see how the server implementation served (or did not serve) the analysis clients employed by those users. This project was done as part of the Committee on Earth Observation Satellites (CEOS) Working Group on Information Systems and Services (WGISS) project activity, the WGISS Test Facility for CEOP.

1. The ability of analysis application software, especially GrADS, to communicate with the gateway:

a) length of WCS request URL

Coverages provided by WCS servers that serve high data volumes are generally based on the physical data files in server's databases or file systems. That is, a server offers one coverage for one physical file. Our MODIS server is implemented in this approach. Because one MODIS file usually contains more than one scientific parameter, additional parameter identification is needed after a physical file name, which itself can be quite long due to inclusion of date and time. This results in a coverage name dozens of characters long, e.g.:

MOD05_L2.A2004209.0230.005.2007025064844.hdf:Grid:mod05:Water_Vapor_Near_Infrared.

The long coverage name plus other parameters in WCS getCoverage request may make the request URL too long to be handled by some application analysis software.

Note that this is not specifically a WCS problem, but can become one when WCS is a component of distributed workflows.

b) encoding of special characters in URL

In addition to the length of URL, special characters included in coverage identification may also cause problem in application software, for example, characters '&', ':', and '%' in coverage identification may cause failure of the application software. Again, this is not specific to WCS per se, but is a potential factor when WCS URLs are embedded in distributed workflows for pass-through.

c) use of information included in the response, especially the lineage information (see lineage issue)

In addition to numerical coverage data, a returned coverage most likely includes additional information to describe the numerical values. The most important pieces of

information include georeferencing of data points, scientific meaning of data values such as measurement/parameter type and units, and provenance information. Among these pieces of information, georeferencing and units are usually easy to understand in the CF-netCDF and the HDF-EOS encoding formats, although HDF-EOS does not have strict rules on units as CF-netCDF does. The meaning of measurement/parameter is often expressed by the name of the variable data array, such as Water_Vapor_Near_Infrared. This meaning is usually not machine understandable and will need additional ontology or taxonomy to describe. In our use cases, we assume that the human users of the application software will understand the parameter he/she is requesting and further machine understanding of the parameter meaning is not particularly important.

Thus, the most difficult issue is to provide provenance information that documents the processing steps from input data sets to output coverage, although here we do not attempt to address provenance information other than server side processing such as those related to observation/measurement, retrieval/modeling, and processing steps leading to the creation of the input data sets. Because of its broadness and importance, we will write a separate technical note to discuss the provenance information.

2. Coordinate reference system (CRS) transformation

a) The necessity of CRS transformation

Many of the data products served are in swath data whose values are geometrically aligned with satellite swath coordinate system, while data users usually need data in earth coordinate, geographic or projected, reference system. CRS transformation (CRST) is therefore necessary in the gateway. Allowing users to request CRST in a standard way is one of the key advantages that the OGC WCS protocol provides. In the broad sense, CRST includes any change of the CRS associated with a dataset, including transformation from satellite swath CRS to geographic/projected CRS and various reprojections among projected CRSs. Our user community (i.e., the CEOP scientists) is mostly interested in using the satellite data in geographic latitude/longitude coordinate reference system. Thus, the CRST implemented in our server is mainly swath to geographic CRS transformation.

b) The WCS standard of describing CRST

The WCS specification provides a very simple standard parameter in defining how CRST should be done, i.e., CRS. A server lists all the CRSs it can offer in an XML element named SupportedCRS and a client use "CRS=crsValue", where crsValue must be one of the values listed in the SupportedCRS element, to specify a desired CRS. Although CRST itself is a kind of geometrical processing of data, it is related to the nature, or scientific meaning, of a data product being served because change in CRS will involve derivation of new data values from values in an input dataset through interpolation and/or resampling. WCS version 1.0.0 allows clients to use the InterpolationMethod parameter to specify a method to be used in deriving new values in desired CRS from input values in input CRS. Three classical digital image processing interpolation methods, namely nearest neighbor, bilinear, and cubical convolution, are prescribed in WCS1.0.0. A server may offer one or more of these methods and a client can select any one of the offered methods.

c) Implementation options

While the WCS protocol defines CRST and related parameters, different servers may select different approaches to implement a CRST. There are usually two ways to perform satellite swath to geographic CRST. One is a forward method in which the data points in swath CRS are mapped to the output geographic CRS and interpolation is performed in the output coordinate system. The other is a backward method in which

the data point coordinate values in the output grid are mapped backwards to the input swath coordinate values and interpolation is performed in the swath CRS. In our implementation, we used forward mapping and nearest interpolation during the forward mapping. The reason to use forward mapping is that there is not a straightforward way to find a swath coordinate values from a geographic coordinate values. In our earlier implementation, we used a polynomial fitting function to derive swath coordinate values from geographic coordinate values. The polynomial fitting function, usually with an order not exceeding 3, performs fast than point-to-point forward mapping but lacks accuracy at the edge of a swath, unless multiple fitting functions are used, which, however, will result in additional mosaicking processing at the boundaries of the valid regions of different functions.

In addition to the coordinate transformation and interpolation, the output grid point value may also involve further aggregation, as oppose to resampling, when the request grid cell size is larger than the input swath cell size. The algorithm is product specific, i.e., depending on the nature of product data value. The most common approach is to average the multiple input data points falling into one output grid cell, but there are also others such as picking the majority (mode), maximum, or minimum value. Currently all of our offered coverages are ratio data types and we used averaging method to perform aggregation. Further implementation will include the majority rule for nominal (categorical) data.

Lessons learned: a) CRST, or georectification/reproject, must be product specific and parameter specific. Parameters in the WCS interface protocol are not sufficient to define the difference among products; b) for server's involving on-the-fly-georectification, limits to spatial and temporal extents often need to be applied due to performance consideration.

3. Mosaicking of multiple file granules

Most currently available WCS servers are offering coverages based on physical files, i.e., each physical file is offered as one coverage or in other words each coverage is associated with one physical file. While this is the most straightforward implementation and reflects clear one to one relationship between a physical file and a coverage, it may not be desirable when a user's area of interest exceeds the extent of one single physical file, in which case multiple requests must be used to get enough data for the interested area and the multiple returned coverage need to be mosaicked at the client side. The alternative is for a server to offer a virtual coverage that does not tied with any one single physical file in the server's file system. The server will dynamically determine which physical file(s) is/are associated with each client request and perform mosaicking, when necessary, of data from multiple files. The advantage of this implementation is that the mosaicking is transparent to clients and the offered coverage can be described in a very general way. The disadvantage is that it may take much longer for a server to respond due to the more complicated processing steps involved in the server side. The client side mosaicking is usually much simpler because the coverages need to be mosaicked are already in the desired geographic CRS. In addition to more complicated processing steps, a server needs to estimate the memory and speed (i.e., space and time) of the server machine in determining how general a virtual coverage can be. Limits on spatial and temporal extents may need to be imposed to make sure the generalization does not exceed server's capability.

We implemented two versions of WCS servers. The MODIS server is a physical-file-based server without mosaicking capability. The AIRS server offers virtual coverage capability where each variable is offered as a global coverage whose spatial extents

covers the entire globe. As mentioned above, such virtual "global" coverage may involve many physical files (240 files per days in our AIRS Level 2 product case). This global coverage was implemented as an alternative to separate, spatially constrained coverages for each CEOP reference site (there are about 30). The reference-site coverage implementation produced scalability problems in the gateway due to the combinatorially large number of resultant coverages (stations * parameters * days).

When a client requests an output coverage with a large spatial and temporal extent, the server may not be able to fulfill the request. To mitigate this, we put a constraint on the temporal extent of less than one day, from 0-hour to 24-hour. Unfortunately the WCS specification does not define how to describe such a constraint in a machine-understandable way. Our server will simply respond with an exception message when a request crosses the 24-hour boundary. The message, in free text, is informative to a human user but not machine parsable. In addition to that on the temporal extent, constraints on spatial extent can also be applied. We used temporal constraint because our user's study areas, the CEOP sites, are globally distributed and allowing a global spatial extent enables users to request data for any CEOP site. When the WCS server is integrated with the OPeNDAP Hyrax server, coverages from individual days are assembled into time series and thus a user can request coverages for any CEOP site and at any time length within the two year (2002-10-01 to 2004-09-30) CEOP observation period.

Lessons learned: It is desirable that a WCS server provide virtual coverage service which involving, at the server's side, on-demand mosaic of multiple physical files. A client need not know the relationship between the offered coverage and the actual physical files in the server. Such server capability, on the other hand, may be at the expense of either server's performance or its spatial/temporal extents, depending on various factors including file size, internal storage and transfer capability, and encoding.

4. Time stamps

The WCS specification provides a time keyword with which a user can define desired temporal extent for a data product. The keyword is flexible enough to allow specifying either time points or time periods. It is expected that the output coverage from a server should exactly reflect the time requirement requested by a user. For example, when a requested time is accurate to a minute, the values in the output coverage should reflect the measurement of that specific minute, and when the time is accurate to a day, the value should be a daily measurement. In actual use scenarios, however, things may become more complicated, depending on the data product being served. For gridded high level products such hourly, daily, and monthly accumulated precipitation, the server can offer unambiguous time information and a user can obtain coverage with temporal extent matching exactly with what is requested, such as to specific hour(s) and day(s). In case of ungeorectified swath data, approximation and customization in terms of time may be necessary.

In our Level 2 AIRS products, our server offers "daily" coverage and a user can request data for a specific day. However, the observation times for different points in the coverage are not the same and thus coverage values do not actually represent a daily measurement. An additional data array is needed to record the observation time for each grid point in the returned coverage file. This additional data array may not be easily used by users. It may even not be possible to pack this additional information in a coverage file due to limitations in coverage file encoding format. Therefore it is necessary to discuss with the intended users on how the time information can be best represented and used. Our discussions concluded that it is best to group the

observation into two groups, one representing day time and the other night time. Thus, the output coverage would have a day time component and a night time component, i.e., the coverage data array has a "time" dimension whose size is 2. Initially, we used the average of day time observations and average of night time observations to represent the day and night components of the coverage, respectively. Such average times are not fixed at specific times during a 24-hour period, but rather change with different spatial locations and days. This makes it difficult for our users, who mostly use the GrADS software, to construct time series data for temporal analyses. After further tests, our users suggested that we use two fixed times, one 09:30 and the other 19:30, to represent day and night components of the output coverage.

Lessons learned: a) there are cases where the WCS coverage specification must be understood differently due to special data product characteristics. The temporal and spatial dimensions in WCS are assumed to be perpendicular (or independent). In our swath data product, this assumption is not true. The spatial location of a coverage grid point changes as time changes. A server needs to select a proper granularity to offer such time-variant coverages; b) the server provider should always consult with intended user(s) to decide how to best represent the time values in such output coverages. It is possible that different users may have different requirements and thus different versions of server implementations are needed. This, however, seems not consistent with the goal of the WCS interoperability, i.e., WCS servers and clients should be interoperable at machine to machine level with having to make special arrangements between specific servers and clients. Our exercises indicate that a mutual understanding between service providers and service recipients on product specifics is still required.

Another note about the time dimension in our server: Our current implementation of dividing day and night components is actually not consistent, in strict understanding of WCS protocol, with the offered (and also requested) time description. When a client request one single time, i.e., a day, our server responds with a coverage having a time dimension of size of 2, each for day time and night time, respectively. This may cause problems for more general clients do not have prior knowledge of the returned coverage. An alternative will be to offer two coverages for each day, a day time coverage and a night time coverage. A client will need to send two requests and receive two separate coverage files for a specific day. This will worsen the performance at both the server and the client sides.

5. Quality screening

Quality screening is product specific and a WCS interface protocol is difficult, if not impossible, to define parameters for such screening. A service provider should design specific screening approaches for the product(s) it offers. It may sometimes involve intended users because a specific user may have his/her special requirement on data quality. For example, many of our AIRS variables have different quality levels (e.g., good and best) and different users may have different criteria. At the implementation level, a service provider may also have different alternatives. For example, a server may pack the quality control values into a coverage file for users who want to check the qualities themselves. In our implementation, we perform quality screen at the sever side and do not provide additional QC data to the client. We included in our output coverage all data points that are *not* labeled as "do do not use" . The criterion can be changed, if users request, in server's configuration file.

Lessons learned: we did not get many comments from our users on how quality screening should be performed and how QC data should be packed into the output coverage, partly because our server-side implementation of screening made the process

transparent to them. At the server side, we make sure that data with bad quality are excluded. But in the long term, the adequacy of this solution needs extensive usage in real applications, which unfortunately exceeds the project cycle. With extended maintenance of the server at the NASA GSFC DISC, this user feedback might be obtainable, allowing us to revise our quality approach if needed.

6. Performance and scaling

As previously mentioned, we have two versions of WCS, one providing per-file based service without the on-demand mosaic capability and the other offering virtual coverages that are not tied with specific physical data files. For the per-file based service, the performance is not an issue regarding single request, except for rare cases when a client requests very high resolution output. However, we have internally set certain requirements to prevent such a situation from happening. On the other hand, it is possible that the per-file based server may get a large number of requests during a short period of time (e.g., hundreds or thousands simultaneous requests), in which case it is possible that some requests may time out as the machine hosting the server slows down or runs out of memory or temporary disk space. (We have not yet conducted such scaling tests.) For the version that offers virtual coverage, (note: temporal extent is actually limited by a product's temporal availability in the server's archive), the server is more likely to encounter performance problem because it essentially offers unlimited spatial and temporal extents. Due to the virtual coverage offering, scalability is strongly driven by client's requests. Theoretically, the server should be able to provide a coverage that covers the entire spatial and temporal extents, e.g., a single global coverage from all the available years (2 years in our CEOP use case) at nominal spatial and temporal resolutions (0.25 degree and one-day in CEOP use case). In actuality, our server is not able to deal with such a request and we allow only daily request, primarily due to the number of files needed to access at the server side (240 files per day). Currently the temporal scale-up is provided through the Hyrax server where daily coverages for specific CEOP sites are staged. It is also possible to configure the WCS server to individual CEOP sites and eliminate the need to access all physical files. Our current design is to make the WCS general enough but make use of the scalability of the Hyrax server to meet user's long time series requirement.

Lessons learned: we have not done stress tests to our servers such as sending thousands requests at the same time and requesting a very large coverage having large data volume. However, the staging of daily coverages from the WCS server in the Hyrax server seems to be effective. It can provide users with long term time series in different CEOP sites.

7. Provenance information

A separate technical note will be prepared on this topic given its importance.

8. Database/directory structure and informative URL

Both WCS service instances are currently accessing local physical files in file systems. Directory paths are defined in service configuration files and no database searching/ accessing is involved. If in the future the services are revised to support more products and made general in terms of back end data access, it is possible that database management and remote access will be involved. From the user side, these are mostly transparent. One issue in the per-file based MODIS server is that the physical file path and physical file name are exposed in the offered coverage name to the users. However, the name includes information which is useless or difficult to understand for users, such as the inclusion of both product observation and generation times and version numbers in such file names as MOD05_L2.A2002274.0115.005.2006332104856.hdf. Worse, the

product observation time is somewhat non-deterministic and thus thwarts users attempting to script access to multiple coverages. Our users commented that they prefer service URLs with indicative information such as variable type and spatial and temporal extents. For the AIRS server providing virtual coverage, the coverage is simply named using variable name, such as TSurfAir for surface air temperature. The URLs viewed by the users are constructed in the Hyrax server with indicative information included, such as <http://test.opendap.org:8080/opendap/wcs/CEOP/BRA/ceopL2AIRSTimeOffsetLineage/TSurfAir/2002-10-02.html> in which intended user group (CEOP), location (BRA), variable, and time are clearly presented.

Lessons learned: it is relatively easy and straightforward for the Hyrax server to construct informative URLs for end users when the WCS server supports virtual coverage. The per-file based WCS with actual physical file name exposed is not an ideal implementation. The near term solution to this can be to use a logical versus physical file name lookup table to map physical file name to a more concise and informative logical name and the server can expose a physical file with its associated logical name. For example, the MODIS file name mentioned in the previous paragraph can be replaced using a more informative name such as MODIS_WaterVapor_20021001.

9. Packing ancillary information in output coverage

We mentioned ancillary information, i.e., arrays of per grid point time and quality assurance, in the previous discussion. There are potentially more ancillary fields including, but not limited to, viewing and illumination geometry arrays, cloud percentage, standard deviation, and measurement/retrieval error value. Whether such information should be included in the output coverage is dependent on users' specific needs. A server has different options to expose such information to users, such as in coverage description and metadata. The most straightforward and simplest way for the server is to offer these ancillary data arrays as coverages in their own right, just as the main variable coverage. For example, a server may offer land surface temperature and its associated cloud fraction as two separate coverages. A user needing cloud fraction to evaluate the land surface temperature may want to acquire both coverages. This implementation can avoid potential difficulties in data encoding because some data encoding formats do not allow multiple data arrays to be packed into one single file.

Lessons learned: we currently do not include ancillary data arrays in output coverage. The output coverages in our service are already screened using ancillary data information, particularly the quality control arrays. When required, the WCS server can be revised to also offer ancillary data coverages so that a user will be able to request such coverages. A potential issue for such implementation is that it will require user to link an ancillary coverage with the actual variable coverage, which may not be an easy task when there are many variables and many ancillary coverages, especially when the ancillary coverages are not simultaneously requested with the variable coverage. In contrast, it will be much easier if all related ancillary data are packed with the variable data in one output physical coverage file. Even in this case, a number of additional issues arise, such as where in the payload to include the information where it can be discovered and accessed, as well as how to translate into standard or conventional units.