Grooming Big Data from Afar

Peter Baumann

Jacobs University
Bremen, Germany

p.baumann@jacobs-university.de

Alex Mircea Dumitru

Jacobs University
Bremen, Germany

m.dumitru@jacobs-university.de

Vlad Merticariu

Jacobs University
Bremen, Germany

v.merticariu@jacobs-university.de

ABSTRACT

Flexible, scalable services on massive geo data receive much attention today. In particular, the OGC Web Coverage Service (WCS) standards suite has established a best practice for versatile access and retrieval on spatio-temporal “Big Data”. Fewer efforts have been devoted, though, to an easy-to-use, standardized way of maintaining a service’s offering. Our experience from supporting a series of heterogeneous, large-scale services reveals that this can become tedious indeed, due to heterogeneity and incompleteness of incoming data, operator-less transformation and ingest of large amounts of files, as well as the need for narrowly focused manual corrections and updates sometimes.

In this contribution, we present the WCS-T (for “Transaction”) specification which enables users and machines to perform atomic insertion, updating, and deletions through simple Web requests. WCS-T has been implemented and is actively being used in projects; recently, it has entered the adoption process in OGC to become part of the WCS suite.

Categories and Subject Descriptors

H.2.8 [Spatial databases and GIS], H.3.2 [Information Storage], H.3.5 [Web-based services]

General Terms

Standardization

Keywords

geo services; datacube; OGC; coverage; Web Coverage Service; standards.

1. INTRODUCTION

“Big Data” discussions usually focus on gaining insights from data sets which are massive, complex, or otherwise hard to evaluate. This silently assumes that a dataset as such – at least when collected and stored in a data center – is homogenized already. Little attention is given to maintaining data offerings in a way that they are accessible. If such questions are addressed then usually with respect to long-term data preservation [7], processing of data into higher-level products can become tedious, due to

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author. Copyright is held by the owner/author(s).

SIGSPATIAL’15, November 03-06, 2015, Bellevue, WA, USA
ACM 978-1-4503-3967-4/15/11.

<http://dx.doi.org/10.1145/2820783.2820854>

heterogeneity and incompleteness of incoming data, operator-less transformation and ingest of large amounts of files, as well as the need for narrowly focused manual corrections and updates.

In this paper, we present the WCS-T (for “Transaction”) specification. Our contribution consists of a high-level service definition which enables users and machines to perform atomic insertion, updating, and deletions through simple Web requests coherent with the OGC WCS suite of standards [3] [8]. WCS-T has been implemented and is actively being used in projects; recently, it has entered the adoption process in OGC to become part of the WCS suite.

An earlier specification for imagery (i.e., gridded coverages, see Section 2 for details) updates has been presented by Whiteside, standardized by OGC as WCS 1.1.4 [10]. While this has shown necessity and feasibility of a standard for updating service offerings it suffers from several drawbacks. The most important one is that it has been established before the corresponding OGC coverage standard [5] and, therefore, is not aligned to it. Further, it focuses on the bundling of sets of updates into a transaction (hence the name), but does not detail the semantics of updates sufficiently; for example, it is left unclear how a 3-D datacube can be updated with a 2-D slice (e.g. adding an image to a timeseries).

This has led to establishing a specification for WCS-T 2.0, which is presented in this paper. At the time of this writing, the candidate standard is in the adoption process of OGC and available publicly [4]. To the best of our knowledge there is no other approach published which offers versatile update functionality through easy-to-use Web requests.

2. BACKGROUND: COVERAGES

The Open Geospatial Consortium (OGC) develops and maintains open standards for spatial data and services, based on the special expertise of over 500 members from academia, industry, and agencies. This is pursued in tight collaboration with related bodies like ISO, W3C, OASIS, and others.

2.1 OGC Coverages

OGC has established a canonical coverage implementation model [5] from ISO 19123 which is concrete, concise, conformance testable down to pixel level, and therefore interoperable. This coverage model is capable of expressing multi-dimensional sensor, image, simulation output, and statistics data. Examples include 1-D timeseries, 2-D satellite imagery, 3-D x/y/t image timeseries and x/y/z geophysical datacubes, as well as 4-D ocean and climate data.

In terms of data structure, a Coverage is a subtype of the general class Feature. Ignoring the (rarely used) coverageFunction, a Coverage consists of four main components. The *domain set*

describes the locations (in ISO 19123: “direct positions”) for which values are provided. The *range set* contains these values. The *range type* describes the coverage's range set data structure using the concepts of SWE Common [9]. Finally, a domain-extensible slot for metadata is provided. Data kept there are not interpreted by WCS as such, but they are delivered. The intended use is to define concrete metadata structures and their semantics in extensions or application profiles, such as EO-WCS [7].

Notably, this coverage definition is completely format agnostic; although UML and GML are used for the global and detailed structure definitions, respectively, and implementation does not need to support GML as a format. For example, a coverage service only delivering GeoTIFF and NetCDF is perfectly acceptable and can pass the conformance tests.

2.2 The OGC Web Coverage Service Suite

WCS Core consists of only the most fundamental functionality: extract a coverage (or a subset thereof). Depending on the formats supported, the coverage result can be delivered in some suitable encoding format.

Two terms will be important in this context. A coverage's *Native Format* is the format in which it is stored on a server. A coverage's *Native CRS* (Coordinate Reference Systems) is that (well-defined spatio-temporal) CRS in which the stored coverage's coordinates are expressed.

This is instrumental to achieve one of the essential properties of WCS Core: to always deliver coverage data unaltered in whatever way. Only by consciously choosing a lossy format, or one of the interpolating extension operations below, range values possibly get altered through numerical inaccuracies or approximations during format encoding.

The property of always delivering the exact data in naive requests is a fundamental difference to WMS which performs approximation and rendering (i.e. changing) of values in the process of generating a visual representation. Therefore, WMS is designed to form the end of a processing pipeline (where a human consumer sits) whereas WCS can be deployed anywhere at the beginning, inside, or at the end of such a pipeline.

WCS Extensions add further functionality facets. The WCS Range Subsetting Extension allows extraction and recombination of bands (“channels”, “variables”) from some multi-band (e.g., hyperspectral) coverage. The WCS Scaling extension provides functionality for changing a grid coverage's resolution. Transformation of coverages, as well as input subsetting bounding boxes, across different CRSs is enabled by the WCS CRS Extension. Control over a server's interpolation behavior is ensured by the Interpolation Extension.

3. UPDATING COVERAGES

WCS-T is an extension to WCS enabling service providers to update (or allow updating) of coverage offerings. All requests must be atomic: any server-side modification must either succeed completely, or have no effect at all. In fact, the ACID (atomicity, consistency, isolation, durability) properties of transactions in databases have motivated the title “Transaction”.

A major design criterion was service symmetry in the sense that the output of a *GetCoverage* request can be fed into a WCS-T request for insertion or updating without any alteration or amendment. This is an indispensable prerequisite for larger – in particular: machine-to-machine – service mashup.

WCS-T defines three new request types. *InsertCoverage* creates a new coverage from the input data. *UpdateCoverage* modifies a coverage through a choice of different techniques. *DeleteCoverage*, finally, removes one or more coverages from a service offering. We inspect each in turn; for ease of reading we use the GET/KVP protocol binding but mention in passing that the same functionality is available through several other protocols.

3.1 Insertion

With the *InsertCoverage* request, a new coverage is established on the server and available for subsequent retrieval, e.g. through *GetCoverage*. The coverage can be provided as part of the request (this is obviously not possible through a GET/KVP request, but through POST/XML or SOAP, for example) or by reference through a URL from where the server can retrieve the coverage. xlink references may also be used inside the coverage to reference some constituents (like range set) rather than providing them verbatim. All references must be resolvable by the server.

The type of the new coverage is determined from the input coverage provided. For example, if no georeferencing is supplied then the new coverage on the server will be a *GridCoverage*; if it is georeferenced with a regular grid underneath then a coverage of type *RectifiedGridCoverage* will be created.

One problem is related to the identifier of the new coverage. These identifiers must be unique within a server's offering, but often a client will not know what identifiers are used already. To this end, the server by default will use the identifier provided in the input coverage (if any). Alternatively, the optional parameter *USEID*, with a value of “new”, directs the server to create some unique identifier (thereby ignoring any identifier provided). In any case, the final identifier of the new object is returned back to the caller. A complete request might look like this:

```
http://acme.com/wcs
? SERVICE=WCS & VERSION = 2.0
& REQUEST=InsertCoverage
& COVERAGEREF=http://bcme.com/archive/hurricane.nc
& USEID=new
```

3.2 Deletion

Deletion of coverages is done by simply listing their identifiers. After successful termination of the complete request none of the coverages will be accessible on the server any more. Their identifiers may be reused; any such coverage will bear no relation to the one previously deleted. Here is a sample request:

```
http://acme.com/wcs
? SERVICE=WCS & VERSION=2.0
& REQUEST=DeleteCoverage
& COVERAGEID=cov1,cov2,cov4,cov8,cov16
```

3.3 Update

The *UpdateCoverage* request modifies all or part of a coverage existing on the server. While this sounds simple, a main difficulty in the design of this operation was to determine the admissible changes, as the coverage's internal consistency must be preserved. This excludes many constituents from getting changed, such as the coverage's cell (range) type, number of dimensions, and CRS.

UpdateCoverage has been designed to allow the following changes:

- A cell value in the range set may be replaced by a new value. For example, in an RGB image a pixel may be

superseded with new values for the red, green, and blue channel, resp.

- A range component of a cell value may be replaced selectively. For example, an *UpdateCoverage* request on an RGB image may update only the red band.
- An arbitrary set of cell values in the coverage's range set may be updated this way.
- If marked as extensible, upper and lower bounds of the bounding rectangle establishing the domain set may be extended or reduced.

Concretely, the *UpdateCoverage* operation works as follows. A coverage's range set can be replaced completely or partially (so-called "partial update"). For a complete update, no further parameters are required beyond the coverage addressed; in this case, the update request is as simple as the following:

```
http://acme.com/wcs
? SERVICE=WCS & VERSION = 2.0
& REQUEST=UpdateCoverage
& COVERAGEID=CoverageToBeUpdated
& COVERAGEREF=http://bcme.com/archive/hurricane.nc
```

For a partial update, the set of target positions to be updated must be indicated by specifying one of the following optional parameters:

- The domain subset to be updated, expressed in the stored coverage's Native CRS; by default, the whole range set gets replaced. If the coverage is non-extensible then the replacement bounding box must lie completely inside the coverage's domain set. If it is extensible then the bounding box may be inside, outside, or partially overlapping. If an extension takes place then the new domain set is given by the smallest common box around the union of original domain set and the update box. Cell values in the result extent which were previously not existing are set to some nil value, if defined with the coverage; if not, values are set to 0.
- The list of range components to be updated; by default, all range components are replaced. For example, an *UpdateCoverage* request may update some hyperspectral image through an RGB image of which only the red band is used for updating while leaving all other bands unaltered. As range components are not necessarily named identically (for example, an RGB image for update might use *red*, *green*, *blue* while the Landsat image to be updated refers to *band1*, ..., *band7*) it is possible to indicate a mapping. In our example, this mapping would be written as *red:band4,green:band3,blue:band2*.
- A mask may indicate which direct positions of the input coverage are to be used for updating the coverage on the server. Only those direct positions are considered for the update where the mask overlaps with the coverage and where additionally the value of the mask at this direct position has a value of 1. As with the input coverage, the coverage representing a mask can be provided dir-

ectly with the request or per reference through a URL that the server can access

Let us consider an example. A 2-D lat/long input coverage may replace a rectangular part of a 3-D lat/long/t coverage (e.g., a satellite image timeseries), given by some 3-D bounding box [*lat1:lat2, long1:long2, t1*] indicating a slice at time position *t1* with the Lat / Long trimming extent indicated. Cells outside of this domain will remain unaffected. Further, assuming a hyperspectral range type, the input coverage should substitute only the red band, leaving all other bands unaffected. Finally, a mask may be provided indicating those areas to be updated by a value of 1. The corresponding request may look as follows:

```
http://www.acme.com/ows?SERVICE=WCS&VERSION=2.0
& REQUEST=UpdateCoverage&COVERAGEID=C001
& INPUTCOVERAGEREF=
    http://www.acme.com/update.nc
& SUBSET=Lat(5.0:10.0)
& SUBSET=Long(20.0:30.0)
& SUBSET=H(0.0)
& MASKREF=http://www.acme.com/mask.nc
& RANGE COMPONENT=red:band4
```

3.4 Maintenance Recipes

As different research datacenters hold heterogeneous datasets in various formats (e.g. GeoTIFF, NetCDF, HDF etc) and of various degrees of completeness with regards to the needed information to describe the coverage, we have created an application based on the WCS-T standard that allows for flexible, reproducible and safe ingestion and maintenance of the coverages.

The application introduces two concepts to help researchers model their ingestion process better: recipes and ingredients.

The recipe describes the method of translating and modeling the current data archive into one or more coverages by describing the translation of the raw data into a coverage and the techniques to retrieve any other relevant metadata. For example, a dataset of satellite images over the same area at different points in time would consist in its raw form of a set of geo-referenced 2-D raster files, with the date of acquisition encoded either in the filename or inside the file (e.g. tiff tag). However, this dataset would best be represented as a 3-D timeseries in a WCS endpoint, allowing users to both retrieve individual slices in time and to analyze a certain area's evolution through time. A timeseries recipe does exactly this, models the initial 2-D files into a 3-D coverage.

Each recipe consumes an ingredients file that contains all the parameters needed to "cook" the recipe, some common for all recipes (e.g. the WCS endpoint, the paths to the files that compose the dataset etc), others specific to the recipe (e.g. the tiff tag containing the acquisition date for a timeseries recipe). The ingredients information is represented in JSON format, allowing for a simple and extensible structure that can accommodate any recipe type.

All the information the user needs for reproducing the ingestion is the ingredients file, which can be stored under version control or distributed together with the data. This makes sharing ingestion information a breeze for data providers.

4. IMPLEMENTATION

Rasdaman [2] [6] is being used for assessing all WCS specifications prior to their standardization; naturally, therefore, the system is WCS Core Reference Implementation and also

implements all WCS Extensions. Rasdaman databases of individual sizes exceeding 100 TB are up and running, and complex array queries have been distributed successfully across more than 1,000 cloud nodes [6].

As an Array Database System, rasdaman supports bare arrays in a domain-independent manner. The geo semantics of WCS coverages is maintained outside of the rasdaman server (*rasserver*), by a separate component. The range set is stored as an array maintained by rasserver whereas technical metadata tables store domain set, range type, etc., queryable through regular SQL. Notably, rasserver can read data sitting in some database system (such as PostgreSQL), but it can also maintain its storage-optimized structures in the file system, and additionally can read from any pre-existing archive structure.

With the help of the geo metadata, incoming OGC requests are translated into rasserver array queries. This architecture allows rasdaman to expose OGC services while taking full advantage of the optimizations and scalability offered by the array processing engine. In the same spirit, the Java based WCS-T implementation relies on a servlet receiving client requests, which, after validation, are translated into rasdaman database queries. Each request is wrapped in a transaction to ensure ACID properties.

Performance-wise, most of the time is spent applying the operation on the array. This is the case because, usually, the actual array data is orders of magnitude larger than the geo-semantics metadata. Given the same hardware capabilities, there are several factors influencing the ingestion time: data format in which the coverage is given (e.g. a 3D coverage might be stored in several 2D GeoTiff files corresponding to different slices, or in a single NetCDF file with variables corresponding to different timestamps, case in which some preprocessing might be required before ingesting), data dimensionality, pixel data type, whether the data is ingested or only referenced.

Since most of the data sets that we have ingested so far using WCS-T have been 3-D cubes provided as 2-D GeoTIFF files with the same Latitude / Longitude bounding box, corresponding to different time slices of the same coverage object, we have measured the import time relative to the size of the dataset. Table 1 shows an overview of the measurements obtained. Although a comprehensive evaluation is still on the agenda, the speed of ingestion overall appears encouraging.

Table 1. Data ingest times through WCS-T

Dataset	Size (GB)	Import time (min)	In-situ
Australia water quality I	1	1	No
Australia water quality II	100	80	No
ESA Landsat database	1000	16	Yes

5. CONCLUSION AND FUTURE WORK

WCS-T is the latest component in OGC's WCS "Big Geo Data" suite. The adoption process is expected to finalize within 2015.

WCS-T offers several significant advantages. First, it simplifies maintenance of any service offering coverages. This is due to the standardization of a Web-based interface, as opposed to the traditional way of homegrown individual scripts. At the same

time, this eases consistency of update operations due to the clear definition of prerequisites, parameters, and postconditions. Moreover, WCS-T paves the way for mashups of automatic exchange of original coverage data or derived products where simple scripts can extract coverage information from one server and inject it into another.

In parallel to development of the specification, WCS-T has been implemented as part of the open-source geo analytics server, rasdaman [2] [1]. Meantime it is the standard tool for data ingestion by ourselves and rasdaman service operators. Specifically, the EU supported EarthServer initiative, currently in its second phase, is establishing several Petabyte-sized 3-D and 4-D datacubes for dynamic mix-and-match. An easy to handle, automated ingestion and maintenance facility is indispensable, and the benefits of WCS-T will be leveraged at a large scale.

6. ACKNOWLEDGMENTS

Research has been supported by the European Commission under H2020 grant EarthServer-2.

7. REFERENCES

- [1] P. Baumann. A database array algebra for spatio-temporal data and beyond. In Proc. NGITS99, number 1649 in LNCS, pages 76–93. Springer, 1999
- [2] P. Baumann: On the Management of Multidimensional Discrete Data. VLDB Journal 4(3)1994, Special Issue on Spatial Database Systems, pp. 401 - 444
- [3] P. Baumann: Web Coverage Service (WCS) Interface Standard – Core, version 2.0. OGC 09-110r4; <https://portal.opengeospatial.org/files/09-110r4>; seen 2015-06-20
- [4] P. Baumann: OGC WCS Extension – Transaction, version 2.0. OGC 13-057. http://external.opengeospatial.org/twiki_public/CoveragesDWG/CoveragesBigPicture, seen 2015-06-20
- [5] P. Baumann: OGC GML 3.2.1 Application Schema - Coverages, version 1.0. OGC 09-146r2. https://portal.opengeospatial.org/files/?artifact_id=48553, seen 2015-06-20
- [6] A. Dumitru, V. Merticariu, P. Baumann: Exploring Cloud Opportunities from an Array Database Perspective. Proc ACM SIGMOD Workshop on Data analytics in the Cloud (DanaC'2014), June 22 - 27, 2014, Snowbird, USA
- [7] J. Gray, D.T. Liu, M. Nieto-Santesteban, A. Szalay, D. J. DeWitt, G. Heber. 2005. Scientific data management in the coming decade. SIGMOD Rec. 34, 4 (December 2005), 34-41. DOI=10.1145/1107499.1107503
- [8] OGC: WCS standards. <http://www.opengeospatial.org/standards/wcs>; seen 2015-06-20
- [9] A. Robin: OGC SWE Common Data Model Encoding Standard, version 2.0. OGC 08-094r1; <http://www.opengeospatial.org/files/08-094>, seen 2015-06-20
- [10] A. Whiteside: OGC WCS - Transaction operation extension, version 1.1.4. OGC 07-068r4