PUBLICAMUNDI

SCALABLE
REUSABLE
OPEN
GEOSPATIAL
DATA

REPORT FOR

DELIVERABLE D3.1

1 INTRODUCTION

This report provides an overview of the technical characteristics and functionality of deliverable D3.1 “Raster Server”. Its purpose is to provide a short introduction to the software and present the major functionalities and improvements introduced by the PublicaMundi project.

The reader is encouraged to visit the software’s repository (http://rasdaman.org/) to receive:

- Up-to-date versions of the software, along with documentation targeted to developers
- Detailed information regarding all development effort (commits, activity, issues)
- Instructions regarding the installation of the software and its dependencies

2 RASDAMAN

The rasdaman system is a so-called *Array DBMS*, a recent research direction in databases which it actually pioneered. An Array DBMS offers the same quality of service – such as query language, optimization, parallel and distributed query processing – on large multi-dimensional arrays as conventional SQL systems offer on sets.

The conceptual model of rasdaman consists of multi-dimensional arrays with some cell type and n-D extent. The rasdaman query language, *rasql*, adds generic array operators which can be combined freely. Its expressiveness encompasses subsetting, cell manipulation, and general image, signal, and statistical analysis up to, e.g., the Fourier Transform.

The storage model relies on a partitioning of the arrays into sub-arrays called tiles (*elsewhere in literature also known as chunks*). Tiling is transparent to the user, but accessible to the database tuner as an optimization method; for example, for time series analysis tiles would be stretched along time while having a smaller spatial footprint, thereby reducing disk accesses during query evaluation. Storage of tiles is either in a relational database (RDBMS) or in files. An RDBMS – in PublicaMundi this is PostgreSQL – offers the advantage of information integration with metadata plus the wealth of tools available, but comes at some extra cost, e.g., due to the data duplication as well as for transaction handling.

The rasdaman engine as such operates on arrays in a domain-agnostic way. The specific geo-semantics of coordinates, regular and irregular grids, etc. are provided by *petascope*.

Coverages offered by the data service are made accessible over the web by the petascope component of rasdaman, which is also the reference implementation of the WCS and WCPS standard. Petascope consists of Java servlets that leverage several open source geospatial and geometry libraries, as well as rasdaman data access libraries and relational database access components. Basically it translates incoming processing requests (*be it a WCPS query or a WMS request*) into rasdaman (rasql) queries to efficiently fetch and process array data (according to information in the coverage rangeType and domainSet elements). It then translates the output into the proper coverage type, formatted according to the requested encoding. Moreover, the services allows for delivering coverage data in other formats (not necessarily maintaining all coverage metadata) such as GeoTIFF or non-georeferenced image formats such as PNG, which are suitable for direct image display.

3 CONTRIBUTIONS IN PUBLICAMUNDI

3.1 RASTER STORER

The PublicaMundi project front-end is based on CKAN, an open source data publishing platform that streamlines the process of publishing, sharing, finding and using data. In this context, several CKAN extensions were developed to allow for ingestion of georefferenced data, be it vector or raster. In the D3.1 deliverable, our task was to develop a raster storer plugin for CKAN that automatically ingest data into rasdaman and exposes it through a variety of OGC services.

The work performed consisted of two separate work items, implementing the WCS-T standard to allow simple ingestion of geodata into rasdaman, and creating a CKAN plugin that mediates the flexible user formats allowed by our front-end with the more rigorous format accepted by WCS-T.

As the WCS-T standard has its own section in this document, we will only focus on the CKAN plugin here. The plugin receives a dataset from the user, containing geodata in various formats (GeoTiff, JPEG2000, zip archive etc) and using the GDAL library reads or deduces all the necessary metadata directly from the file and creates a GML file based on it that references the original file as the source of the raw data.

This provides an important degree of flexibility. Rasdaman is not directly coupled with CKAN, and allows the use of this ingestion method in other projects that will not use CKAN as the front-end. The user does not have to duplicate metadata information already available in the file; we will read it and massage it in GML format automatically for him, making the publishing process for raster files a bit easier.

Furthermore, data is never copied or moved around, the GML file only references the original file, allowing rasdaman to read its contents and ingest it in its internal format.

The raster storer CKAN plugin was completed and submitted to the github repository, enhancing the CKAN platform with support for raster formats, both in the ingestion and retrieval process. The datasets ingested through this method, automatically become available through all the OGC services that rasdaman offers (WCS, WCPS, WMS).

3.2 WCS-T

Coverage ingestion into the rasdaman engine is done via the Transaction Extension of the OGC Web Coverage Service (WCS). The OGC Web Coverage

Service (WCS) – Transaction Extension (in short: Transaction Extension or WCS-T) defines an extension to the WCS Core [OGC 09-110] for updating coverage offerings on a server. To this end, a set of three new requests is defined:

1. *InsertCoverage* for adding a coverage provided as parameter to the WCS server's coverage offering. After successful completion of this request, this coverage will be accessible through all WCS operations.
2. *DeleteCoverage* for entirely removing a coverage, identified by its coverage id passed in the request, from the WCS server's coverage offering. After successful completion of this request, this coverage will not be accessible through any WCS operation. However, subsequently a new coverage may be created using the same identifier; such a coverage will bear no relation to the one previously deleted.
3. *UpdateCoverage* for modifying parts of a coverage existing in a WCS server's coverage offering, identified by its coverage id passed in the request. All updates must maintain internal consistency of the coverage, as per GMLCOV [OGC 09-146].

All requests defined in the Transaction Extension adhere to the ACID (atomicity, consistency, isolation, durability) concepts of transactions in databases.

The work done for implementing the functionality described in the specification document of WCS-T consisted in creating a new extension module for WCS in petascope (the rasdaman component handling geo-referenced data and offering web services endpoints). The module includes parsers for the new request types and handlers implementing the functionality defined by these.

The complete definition of a coverage is given in the GMLCOV [OGC 09-146] standard. Internally, a coverage consists of two parts:

- rangeSet is the raw coverage data, i.e. an array or list of the point values indicated in the coverage.
- metadata defines all other information about the coverage, such as the coverage id, the coordinate reference system(s) (CRS), the domain information, the range types (also known as bands) and their names, null values, units of measure and many others.

In order to ingest a coverage into rasdaman, the rangeSet and metadata of a coverage are processed separately, in independent modules, and are persisted in an array and a relational database respectively. The request

handler wraps the two processes in a transaction in order to ensure ACID conformance. This implies that, if during the ingestion process, any fail occurs in any of the invoked processes, the server state is rolled back to the one existing before the initialization of the request.

Deletion of coverages is also handled inside a single transaction, even though it consists of different processes: deleting the coverage metadata and deleting the coverage rangeSet.

Currently, rasdaman supports processing of two of the three request types defined by WCS-T: InsertCoverage and DeleteCoverage. These two operations allow storing and deleting, without limitations, of coverages adhering to GMLCOV [OGC 09-146]. While processing of UpdateCoverage operation is not yet supported, an equivalent operation can be achieved by deleting a coverage and inserting a new one with the same coverage identifier.

The main challenges in implementing the rangeSet processing module consisted in identifying a valid datatype for the given tuple list, as the GMLCOV standard doesn't require explicit mentioning of the data type in the coverage definition. This has been overcome by inspecting the coverage dimensionality and range types, which yield the structure of the data type (for example, a coverage having a RGB image as rangeSet will result in a structured type, having three components). Further, in order to determine the base type of each rangeType component, the gdal library is being used. The base type is fed into the structure determined at the previous step, yielding the final data type for the coverage.

3.3 WMS 1.3

A Web Map Service (WMS) is a standard protocol for serving georeferenced map images over the Internet that are generated by a map server using data from a GIS database. The specification was developed and first published by the Open Geospatial Consortium in 1999. The current version of the standard is 1.3.0.

In version 9.0.4 of rasdaman, the petascope component of rasdaman was supporting the 1.1.0 version of the standard. In the context of the PublicaMundi project, work was performed to support the latest version of the standard and to improve on the overall performance of the offered service.

The work performed was split into two phases, the first one focusing on conformance compliance, while the second focused on the performance of the offered service.

- Conformance compliance. In this phase work was done to ensure that the WMS service that we offer passes the compliance tests developed under the OGC umbrella. The CITE TeamEngine software was run against our implementation and each failed test lead to a fix in the petascope component.
- Performance optimizations. In this phase work was done to improve the request handling speed. Several heuristic improvements were developed, based on previous papers and experiments in the context of the rasdaman project. One of these is the pyramid levels optimization where for each map layer a set of scale levels are precomputed and stored in the database allowing for instant retrieval time for clients requesting these layers at specific scale factors. As each WMS request is translated into a rasql query to retrieve the resulting data, several enhancements were made on the translation process, especially when applying styles over existing layers. The two are now always combined into a single query, eliminating the processing step done on the petascope level.

The current implementation of WMS in rasdaman passes the OGC conformance tests for the latest version of the WMS 1.3 standard. Furthermore, the service is now more reliable and offers better performance than the previous implementation of the standard. Tests are currently being performed to correctly measure the performance increase and will be presented during the review meeting.

3.4 RASDAMAN COMMUNICATION PROTOCOL

The goal of this task is to replace the old rasdaman communication protocol with a new, more reliable one.The rasdaman communication protocol ensures the interaction between the principal components of the software:

- rasmanager - component that routes requests to worker servers, acting as a local load balancer
- rasserver - component that executes queries and returns results to the client
- client - responsible with query sending and result retrieval
- rascontrol - component that allows system administrators to dynamically configure a running system.

The previous rasadman communication protocol, which has been a part of rasdaman for more than 10 years, was built using a custom transport mechanism based on HTTP and a custom encoder which used a special

format to encode the fields of a request, ignoring type safety. Therefore, several bugs plagued the system as independent developers added new message types and functionality on an inherently unscalable system. Furthermore due to the hard coupling of the encoding and the transport mechanisms, this component was never refactored in its 10 years life-span as significant work needed to be done with few direct rewards.

In the context of the PublicaMundi project and based on the feedback given by our partners in this project, we replaced this component with one based on the industry standards for message encoding and transport. The new protocol uses Google Protocol Buffers for data serialization, providing a type safe method for encoding which improves the productivity of the developers and decreases the probability of introducing bugs while developing new messages.

The network transfer is realized over TCP using the ZeroMQ library which provides high speed asynchronous I/O over sockets or inter-process for a variety of languages and platforms.

A service based approach was adopted where the Server provides a typed service interface and the Client accesses this service transparently in RPC style. The new implementation provides failure detection on both network and component level.

- *Improved scalability*

  A new server manager was implemented in order to improve the scalability on a single machine. Based on a management strategy, which is read from a configuration file, the following features were added:

  - a minimum number of available servers is guaranteed
  - new servers are started when the load is increased
  - servers are closed if there are too many started and the number of request is small keeping the resource usage linearly dependent to the number of requests

  In contrast with the old implementation, the new server manager never denies client access, spanning new servers if none are available and keeping the use of resources at the minimum.

- *Improved Reliability*

  The old implementation allowed up to 10 clients to connect simultaneously and execute queries, due to a variety of issues and bugs. Current tests, enabled us to connect 80 clients in parallel on a basic quad-core CPU machine with 4GB of RAM and execute queries without

errors, the limitations being imposed only by the OS and the hardware available.

- *Query execution time*

No major improvements were detected during measurements for single query execution, which is an expected result, given the fact that most of the work in rasdaman is done in the processing engine. However, more tests must be run in order to assess the performance gains of the whole framework, measuring the connection, transaction start and other operations that are performed before the query.