Cataloguing and visualizing big Geodata

Final report

Martín Domínguez Durán[1]

1. Wageningen University & Research, Wageningen, The Netherlands

Registration number: 1254246
Period of Internship: 2024-04-08 - 2024-08-08
Date final report: 2024-07-31

Telephone number student: +31651120353

Name of Company: Satelligence B.V.
Host supervisor: Luca Foresta
MGI supervisor: Lukasz Grus

Table of contents

List of Figures

Executive summary

Vivamus vehicula leo a justo. Quisque nec augue. Morbi mauris wisi, aliquet vitae, dignissim eget, sollicitudin molestie, ligula. In dictum enim sit amet risus. Curabitur vitae velit eu diam rhoncus hendrerit. Vivamus ut elit. Praesent mattis ipsum quis turpis. Curabitur rhoncus neque eu dui. Etiam vitae magna. Nam ullamcorper. Praesent interdum bibendum magna. Quisque auctor aliquam dolor. Morbi eu lorem et est porttitor fermentum. Nunc egestas arcu at tortor varius viverra. Fusce eu nulla ut nulla interdum consectetuer. Vestibulum gravida. Morbi mattis libero sed est.

List of abreviations

Table 1: Abbreviation list

Abreviations	**Description**
EUDR	European Union Deforestation Regulation
STAC	Spatio-Temporal Asset Catalog
COG	Cloud-Optimized GeoTiff
OGC	Open Geospatial Consortium
SDI	Spatial Data Infrastructure
S11	Satelligence
K8	Kubernetes
DPROF	Distributed Processing Framework
JSON	JavaScript Object Notation
API	Application Programming Interfaces
HTTP	HyperText Transfer Protocol
SQL	Standard Query Language
FBL	Forest Baseline
DEM	Digital Elevation Map
TCA	Thematic Content analysis

1 Introduction

1.1 Internship organization background

Satelligence (S11) is a company founded in 2016 that specializes in providing satellite-based actionable information by monitoring environmental risks in commodity supply chains and financial investment planning (Satelligence, n.d.). More specifically, the company processes terabytes of satellite imagery to detect environmental risks and presents this information to their clients in a web application to assist them in the migration towards more sustainable sourcing models and the compliance with deforestation-free commodities regulations, such as the European Union Deforestation Regulation (EUDR) (Satelligence, 2023). S11's main focus is continuous deforestation monitoring (CDM) in the tropics using freely accessible satellite imagery. This is a data-intensive task that is achieved by leveraging the benefits of cloud computing, specifically Google Cloud Platform.

1.2 Context and justification of research

Satelligence strongly relies on cloud computing for their services. They process extensive volumes of satellite imagery amounting to terabytes using DPROF, a distributed processing framework created within the company to efficiently process multidimensional spatial datasets. While this processing workflow currently runs smoothly, the company's data and operations teams face challenges when going deeper into the analysis and accessing intermediate results due to the big nature of this data (Satelligence, 2023). Scholars have defined big data as datasets characterized by their high Volume, Velocity, and Variety, which makes it paramount to use advanced processing and analytics techniques to derive relevant insights (Giri and Lone, 2014). In the specific case of Satelligence, their datasets can be categorized as big data due to their: High volume (Terabytes of satellite images processed every day), high velocity (Near – real time processing of these images) and high variety (Imagery coming from different sensors and regions). All these datasets are a specific case of big data: Big Geodata.

1.2.1 Significance of the topic and previous research

In the past decades there has been a rapid increase in the amount and size of geo-spatial information that can be accessed. Nowadays, more than 150 satellites orbit the earth collecting thousands of images every single day (Zhao et al., 2021). This has made data handling and the introduction of spatial data infrastructures (SDIs) paramount when working with such big datasets.

Traditionally, SDIs have served to ease the accessibility, integration and analysis of spatial data (Rajabifard and Williamson, 2001). However, in practice SDIs have been built upon technologies that focus on data preservation rather than accessibility (Durbha et al., 2023). Due to this, an important shift is underway towards more cloud-based SDIs (Tripathi et al., 2020). These platforms need the emergence of new technologies that prioritize seamless access to cloud-stored data, efficient discovery services that ensure the easy location of extensive spatial data, and data visualization interfaces where multiple datasets can be depicted.

Cloud-based data storage

Spatial data, just like any other type of data, can be cataloged into structured and unstructured data. Structured datasets are often organized and follow a specific structure (i.e. A traditional table with rows (objects) and columns (features)). On the other hand, unstructured data does not have a predefined structure (e.g. Satellite imagery and Time series data) (Mishra and Misra, 2017). The management of structured data has witnessed substantial advancements, making it straightforward to handle it systematically using, for instance, relational databases (i.e. With the help of Structured Query Language (SQL)) (Kaufmann and Meier, 2023). In contrast, due to the additional challenges associated with the handling of unstructured data, the developments in this area have taken a longer time to appear.

The emergence of cloud-based archives has been one of the main advancements for unstructured data management during the last decades. In the specific case of geo-spatial data, it has allowed to store terabytes of unstructured data (i.e. Satellite imagery) on the cloud and access it through the network. However, the necessity transmitting data across networks to access it makes it essential to develop new data formats suited for such purposes (Durbha et al., 2023).

At S11, the storage of large geo-spatial data is already managed using Google Storage Buckets, and they are currently in the process of incorporating the conversion to cloud-optimized data formats like Cloud Optimized GeoTIFFs (COGs) and Zarrs in their processing framework (DPROF) to improve efficiency and accessibility.

Cloud-optimized data formats

COG

Cloud-Optimized GeoTIFFs (COGs) are an example of data formats that have been created to ease the access of data stored in the cloud. They improve the readability by including the metadata in the initial bytes of the file stored, storing different image overviews for different scales and tiling the images in smaller blocks. These characteristics make COG files heavier than traditional image formats. However, they also greatly enhance accessibility by enabling the selective transfer of only the necessary tiles using HTTP GET requests (Desruisseaux et al., 2021). Additionally, this data format has been adopted as an Open Geospatial Consortium (OGC) standard. These standards are a set of guidelines and specifications created to facilitate data interoperability (OGC, 2023).

Zarr

Another cloud native data format that has gained popularity recently is Zarr. This data format and python library focuses on the cloud-optimization of n-dimensional arrays. Zarr, differently than COGs store the metadata separately from the data chunks using lightweight external JSON files (Durbha et al., 2023). Additionally, this data format stores the N-dimensional arrays in smaller chunks that can be accessed more easily. While the storage of Zarr files in chunks facilitates more efficient data access, the absence of overviews hinders the visualization of this data in a web map service (Desruisseaux et al., 2021). Due to the increasing use of Zarr for geo-spatial purposes, the OGC endorsed Zarr V2 as a community standard. Nevertheless, efforts are still being made to have a geo-spatial Zarr standard adopted by OGC (Chester, 2024).

Data discovery services

A discovery service that recently has become widely used for the exploration of big geo-data is Spatio-Temporal Asset Catalog (STAC). Through the standardization of spatio-temporal metadata, STAC simplifies the management and discovery of big geo-data (Brodeur et al., 2019). This service works by organizing the data into catalogs, collections, items, and assets stored as lightweight JSON formats (See Table 1.1) (Durbha et al., 2023).

Moreover, there are two types of STAC catalogs: static and dynamic. Static catalogs are pre-generated and stored as static JSON files on a cloud storage. Static catalogs follow sensible hierarchical relationships between STAC components and this feature makes it easy to be browsed and/or crawled by. Nevertheless, these catalogs cannot be queried. On the other hand, dynamic catalogs are generated as APIs that respond to queries dynamically. Notably, dynamic catalogs will show different views of the same catalog depending on queries which usually focus on the spatio-temporal aspect of the data (RadiantEarth, 2024).

Table 1.1: STAC components

STAC components	**Description**
Assets	An asset can be any type of data with a spatial and a temporal component.
Items	An item is a GeoJSON feature with some specifications like: Time, Link to the asset (e.g. Google bucket)
Collections	Defines a set of common fields to describe a group of Items that share properties and metadata
Catalogs	Contains a list of STAC collections, items or can also contain child catalogs.

In the specific case of dynamic catalogs, the concept of STAC API is widely used. In general, an API is a set of rules and protocols that enables different software applications to communicate with each other. In the case of the STAC API, it provides endpoints for searching and retrieving geo-spatial data based on criteria such as location and time, delivering results in a standardized format that ensures compatibility with various tools and services in the geo-spatial community. Moreover, even though STAC API is not an OGC standard or an OGC community standard, the basic requests performed in a STAC API adheres to the OGC API-Features standards for querying by bounding box and time range, returning GeoJSON-formatted results that conform to both STAC and OGC specifications. Ultimately, compared to OGC API-Features, STAC API enhances functionality by providing additional features that users needed (e.g. cross-collection search, versioning) (Holmes, 2021).

Visualization interfaces

The visualization of spatial data brings with it a series of challenges due to its big nature. Dynamic tiling libraries such as TiTiler have tackled multiple of these challenges by creating APIs that dynamically generate PNG/JPEG image tiles when requested without reading the entire source file into memory (*TiTiler*, n.d.). This feature optimizes rendering of images since PNG and JPEG image file formats are more easily transferred through the web.

TiTiler supports various data structures including STAC (SpatioTemporal Asset Catalog), Cloud Optimized GeoTIFFs (COGs), and is currently working on adding support for Zarrs. For the first two the TiTiler PgSTAC specialized extension integrates with PostgreSQL to enhance STAC catalog querying capabilities. For the case of Zarrs, the

TiTiler-Xarray extension is being developed to facilitate the handling of multidimensional data arrays.

1.2.2 Added value of this research

This research aims to identify efficient solutions for the company's current challenges in discovering and visualizing large geo-spatial datasets by integrating cloud-optimized data formats, cloud services, STAC specifications, and dynamic tiling services. The outcomes of this research will: offer valuable insights into the existing data discovery challenges within the company, propose a methodology for integrating discovery and visualization services, and evaluate the effectiveness of dynamic tiling for various cloud-optimized data formats.

1.3 Research questions

- What are the current challenges, practices, and user experiences related to data discovery and data visualization in the company?
- How can cloud-optimized data formats, cloud services and SpatioTemporal Asset Catalog (STAC) specifications be integrated to enhance the process and experiences of discovering big spatial data within the company?
- To what extent do dynamic tiling services can perform in visualizing different cloud-optimized data formats?

2 Methodology

To answer the research questions presented a series of tasks were undertaken. These tasks are presented in the following subsections where they are divided by research question.

2.1 Baseline scenario

The baseline scenario was defined as the set of methods currently being used by members of different teams at Satelligence to find, retrieve and visualize spatial data. This baseline scenario was evaluated qualitatively by interviewing four members of two different teams in the company (i.e. the data and the operations team). To keep a balance regarding experience of the study subjects, both the newest member of each team and a member with at least three years in the company were interviewed.

The questions asked during the interviews were oriented towards two main topics that were covered during this internship: Spatial data discovery and spatial data visualization. For both topics, the questions were divided into questions related to raster and vector datasets. The questions included in the interview can be found in Section 7.1 and were meant to be open questions with multiple possible answers.

Furthermore, based on the answers of the interviewees a workflow was built to represent visually the traditional steps performed to discover and visualize S11 data. This visual representation included estimations of the steps where more time was spent on.

Finally, the answers to the questionnaire were analyzed qualitatively following a Thematic Content Analysis (TCA). This type of qualitative analysis focuses on finding common themes in the interviews undertaken (Anderson, 2007). The extraction of common patterns within the interviews was initially done using a large language model (i.e. ChatGPT 3.5 [openai_chatgpt_2023]) using the prompt presented on Section 7.2. Moreover, the themes identified were further refined based on the interviewer's interpretation.

2.2 Data and service integration

To efficiently integrate tools for big geospatial data discovery and visualization, a series of steps had to be followed. Initially, the datasets were selected. Subsequently, the structure of the catalog was defined. Following this, a Git repository containing the code required to generate the catalog was created. Static JSON files were then utilized to construct a dynamic STAC API. Ultimately, this API was deployed alongside other services using a continuous integration (CI) and continuaus deployment (CD) pipeline. A further explanation of each step is presented in the following subsections.

2.2.1 Dataset selection

Due to the desire of the company to continue moving towards a cloud-based workflow. The datasets that were considered for the catalog, were composed of either COGs or Zarrs. Nevertheless, since some of the data in the company is stored as virtual rasters (VRTs), methods to also index this type of data formats in the STAC catalog were also included. Specifically, S11's long term goal is to store in the catalog datasets that can be classified as follows:

- Static raster data
  - Forest baselines (Stored as COGs)
  - Third-party elevation data (Stored as VRTs)
  - Other static data
- DPROF results
  - Results of continuous deforestation monitoring (Stored as ZARRs)
  - Other DPROF results
- Supply chain data (Vector data)
- Complaince data (Vector data)

Nevertheless, the scope of this internship was limited to raster datasets.

2.2.2 Proposed Catalog structure

The structure of the STAC catalog proposed can be seen on Figure 2.1.

In it a selection of datasets that will be referenced in the catalog is presented and a hierarchical structure composed of thematic collections is suggested. Moreover, the selection of the STAC extensions[1] used for each dataset will be defined in this step.

[1]STAC extensions are additional metadata properties that can be added to a dataset. (e.g. Classes, bands, sensor-type, etc.)

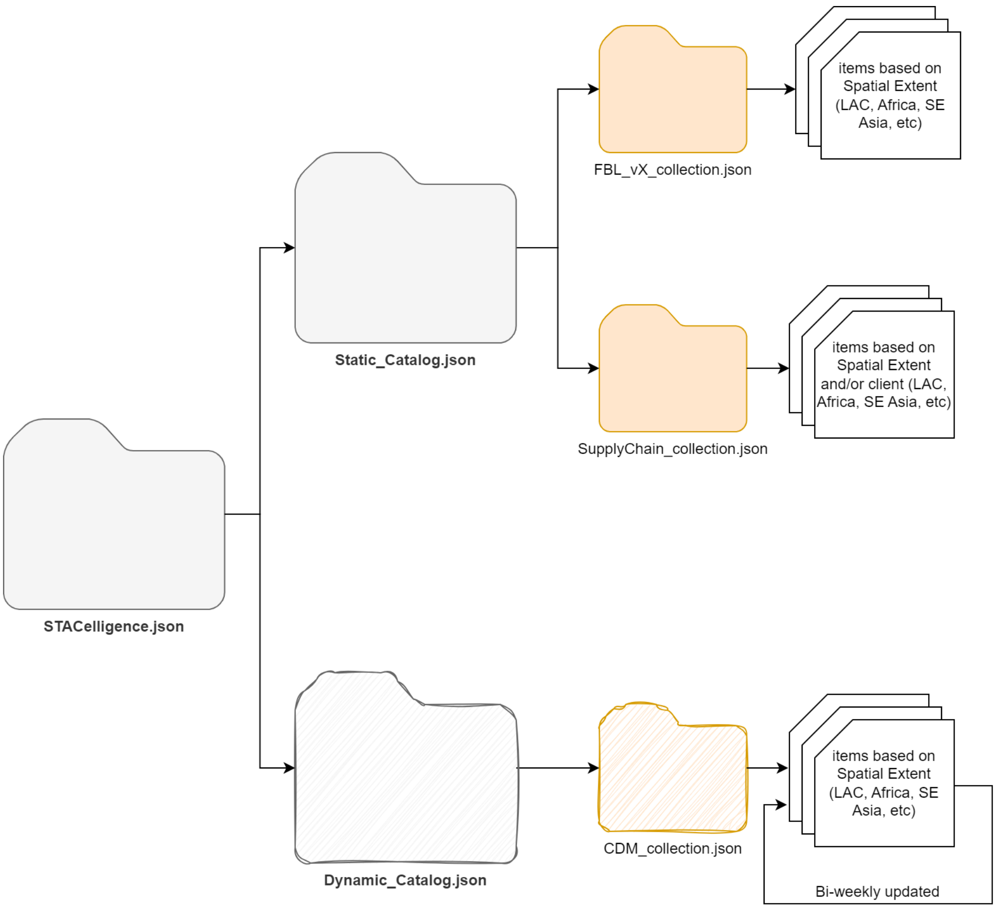

Figure 2.1: Initial proposed STAC structure

2.2.3 S11-cats repository

The s11-cats repository created is composed of a module named `cats` which consists of five submodules described in Table 2.1. Moreover, an overview of the main functionality of cats is presented on Figure 2.2. As seen there, ...

Table 2.1: Description of cats submodules

Submodule	Description
gcs_tools	
general_metadata	
get_spatial_info	
get_temporal_info	
stac_tools	

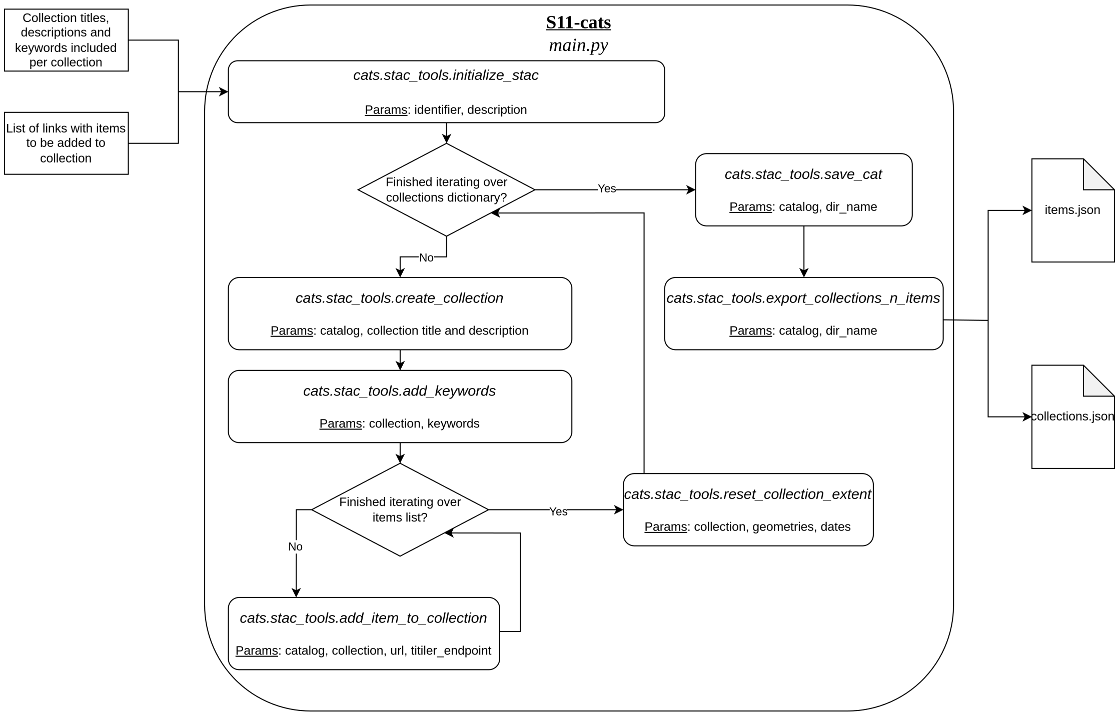

Figure 2.2: S11- cats main function

2.2.4 PgSTAC

2.2.5 eoAPI

- Use of docker containers to run individual applications that can connect to each other.
- deployment of these containers into the cloud.

2.2.6 CI pipeline

2.3 Multi-format data visualization

To assess the performance of dynamic tiling services for visualizing Cloud Optimized GeoTIFFs (COGs) and Zarr data formats, the following approach was undertaken. Firstly, a COG containing forest baseline information for the Riau region of Indonesia was used to create a series of Zarr files, each representing different overviews corresponding to various zoom levels. This preprocessing step, completed by the company prior to the study, ensured that the same data was used across both data formats, allowing for direct comparison. Then, the TiTiler-Xarray service was then customized to work with the specific folder structure of the ZARR overviews previously created. Moreover, containerized versions of both TiTiler-Xarray (for Zarr files) and TiTiler-PgSTAC (for COG files) were deployed locally. The performance was measured by recording the response times for random tile requests at zoom levels ranging from 9 to 18. Finally, to mitigate the influence of cached data on response times, each iteration used a different colormap, with a total of six colormaps employed. This methodology enabled a systematic evaluation of the performance differences between the two data formats in a geospatial data visualization context.

2.3.1 Speed up

The performance of both TiTiler services to dynamically create tiles for the different data formats was evaluated using the Speed Up metric proposed in Durbha et al. (2023) (Equation 2.1). In this case, the Speed Up explains how much did the process of requesting tiles sped up by using a data format A compared to using a data format B.

$$SpeedUp = \frac{t_{formatA}}{t_{formatB}} \tag{2.1}$$

2.3.2 Zoom level influence

Finally, the effect of the level of zoom in a web map visualization on the response times of requesting tiles from the different tiling services was evaluated by fitting an Ordinary Least Squares (OLS) univariate linear regression that followed Equation 2.2.

$$ResponseTime = \beta_1 \cdot ZoomLevel + \beta_0 + \epsilon \tag{2.2}$$

3 Results

3.1 Baseline scenario

3.1.1 Current workflow

The main finding of the interviews were the steps followed currently to discover, retrieve and visualize data. These steps are summarized on Figure 3.1 and show how complex and time consuming the process of discovering and visualizing spatial data can be for a Satelligence employee nowadays. Moreover, the steps followed were categorized in four classes depending on how much time is generally spent carrying out.

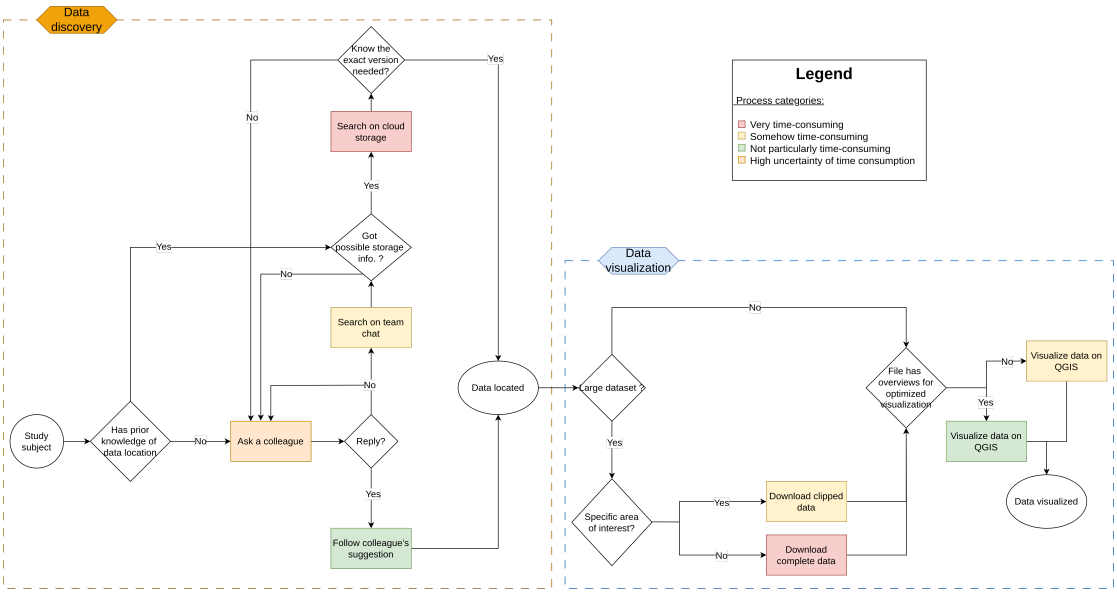

Figure 3.1: Baseline workflow

3.1.2 Main themes found

GPT

- Uncertainty and Dependency on Colleagues

- Sources and Locations of Data
- Data Specificity and Familiarity
- Use of Specific Tools and Methods

Refined themes

The major pitfalls found on the process of data discovery in the company could be summarized in:

- High dependency on colleagues.
- Disorganized structure of Google Storage Buckets.
- Experienced study subjects would find the data location faster.
- Data discovery also dependent on recurrent work with a specific dataset (Google Console highlighted links).
- Not intuitive naming of repositories.
- Not one place where all existing data can be found.
- Big need on going towards a SKI, that is able to provide datasets based on questions like the ones on the questionnaire

3.2 Service integration

Explain here how eoAPI uses multiple services, how each of them helps S11 in their data discovery and vizz tasks, and how did I manage to deploy it

Kubernetes

STAC-API, pgSTAC, TiTiler

3.2.1 Data discovery improvement

Flowchart with STAC

3.3 Performance of multi-format data visualization

TiTiler-PgSTAC & TiTiler-xarray

3.3.1 Raster formats

The comparison of visualization speeds with TiTiler-xarray for Zarr datasets and TiTiler-PgSTAC for COGs are presented on Figure 3.2. In the figure it can be obesrved that COG tiles are requested 2.38 times faster than the same file in ZARR format.

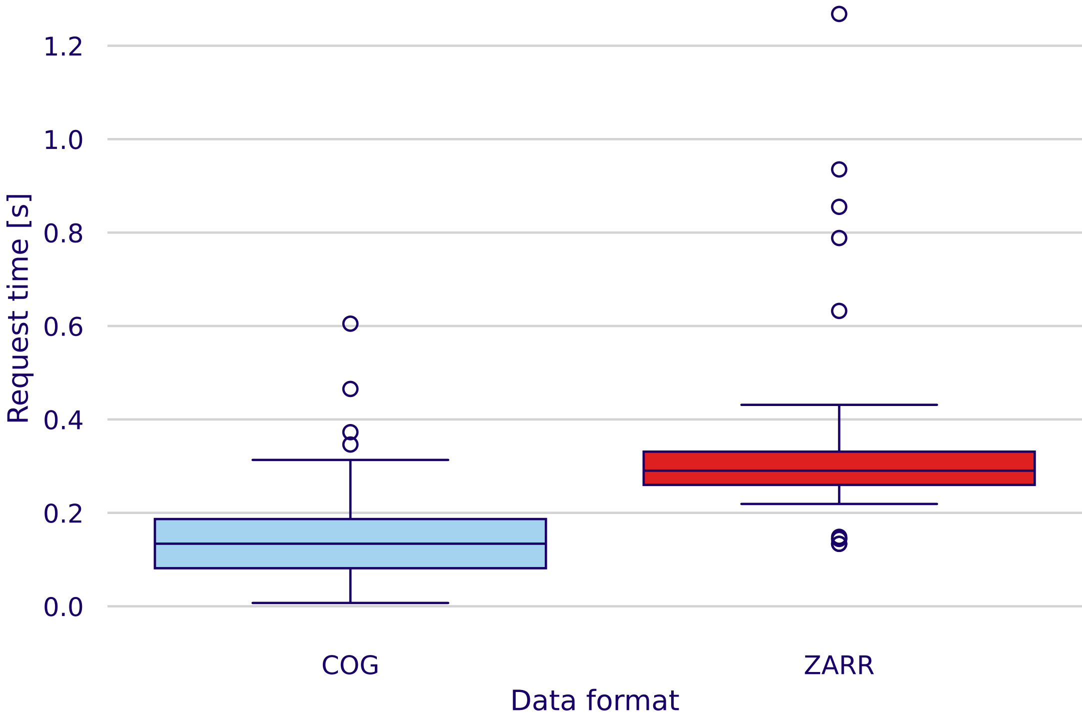

Figure 3.2: Request times depending on data format and zoom level

3.3.2 Effects of zoom level

As seen on Figure 3.3, the zoom level of the map will have an effect on the time spent requesting and getting a tile from a tile server. In this study, it was found that the request times decreased by a factor of -0.013 and -0.005 per zoom level for COGs and ZARRs respectively.

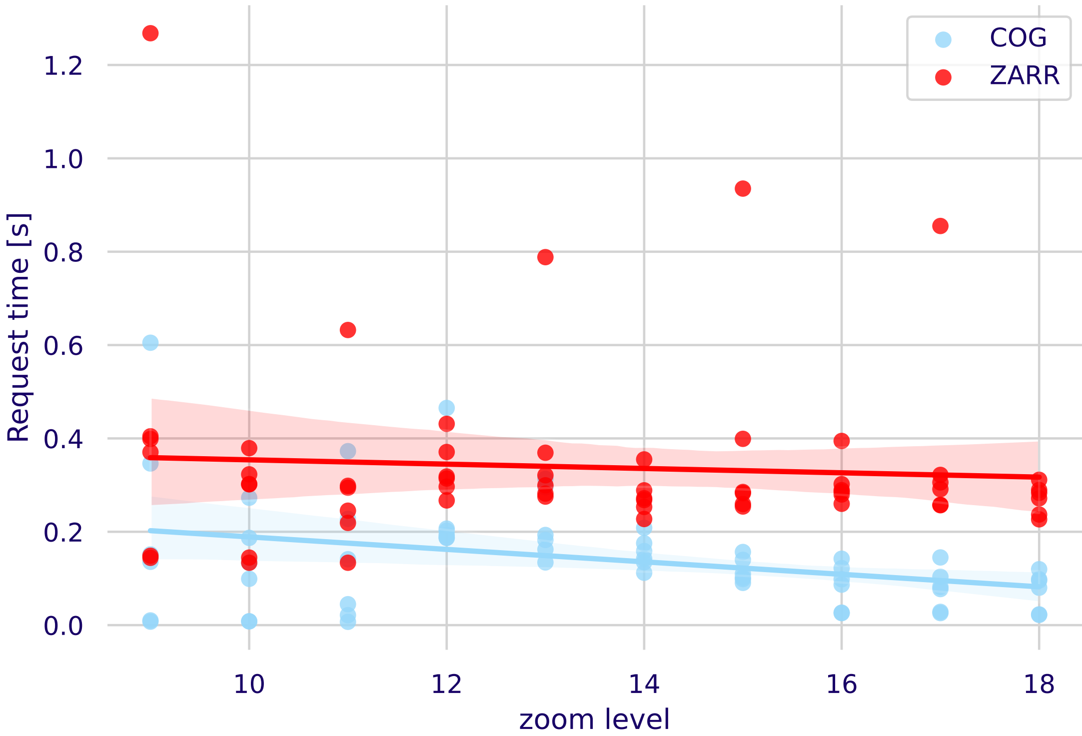

Figure 3.3: Request times depending on zoom level

Talk about how this figure also might indicate that the way the tiles are being called by titiler xarray is not ideal?

4 Discussion

5 Future work

- ndpyramids
- STAC API authentication
- Inclusion of Vector data on STAC

6 Conclusions

References

Anderson, R. (2007). *Thematic content analysis (TCA).* https://rosemarieanderson.com/wp-content/uploads/2014/08/ThematicContentAnalysis.pdf

Brodeur, J., Coetzee, S., Danko, D., Garcia, S. and Hjelmager, J. (2019). Geographic information metadata—an outlook from the international standardization perspective. *ISPRS International Journal of Geo-Information*, *8*(6), 280. p. https://doi.org/10.3390/ijgi8060280

Chester, S. (2024(e)ko Januaryk 18). *OGC forms new GeoZarr standards working group to establish a zarr encoding for geospatial data. Open geospatial consortium.* https://www.ogc.org/press-release/ogc-forms-new-geozarr-standards-working-group-to-establish-a-zarr-encoding-for-geospatial-data/

Desruisseaux, M., Giacco, G., Goncalves, P., Manente, M. and Rouault, E. (2021). *OGC testbed 17: COG/zarr evaluation engineering report.* https://docs.ogc.org/per/21-032.html#toc11

Durbha, S. S., Sanyal, J., Yang, L., S Chaudhari, S., Bhangale, U., Bharambe, U. and Kurte, K. (2023). *Advances in scalable and intelligent geospatial analytics: Challenges and applications* (1st ed.). CRC Press. https://doi.org/10.1201/9781003270928

Giri, K. and Lone, T. (2014). Big data -overview and challenges. *International Journal of Advanced Research in Computer Science and Software Engineering*, *4*.

Holmes, C. (2021(e)ko Januaryk 19). *SpatioTemporal asset catalogs and the open geospatial consortium. Radiant earth insights.* https://medium.com/radiant-earth-insights/spatiotemporal-asset-catalogs-and-the-open-geospatial-consortium-659538dce5c7

Kaufmann, M. and Meier, A. (2023). Database management. In M. Kaufmann and A. Meier (Eds.), *SQL and NoSQL databases: Modeling, languages, security and architectures for big data management* (1–24. pp.). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-27908-9_1

Mishra, S. and Misra, A. (2017). Structured and unstructured big data analytics. *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, 740–746. pp. https://doi.org/10.1109/CTCEEC.2017.8454999

OGC. (2023). *OGC cloud optimized GeoTIFF standard.* https://docs.ogc.org/is/21-026/21-026.html

RadiantEarth. (2024). *Stac-spec/best-practices.md at master · radiantearth/stac-spec. GitHub.* https://github.com/radiantearth/stac-spec/blob/master/best-practices.md

Rajabifard, A. and Williamson, I. P. (2001). *Spatial data infrastructures: Concept, SDI hierarchy and future directions.* http://hdl.handle.net/11343/33897

Satelligence. (n.d.). *Home - satelligence - sustainability monitoring simplified. Satelligence*. Retrieved February 5, 2024, from https://satelligence.com/

Satelligence. (2023). *Internship: Cataloguing and visualizing big geodata*.

TiTiler. (n.d.). Retrieved February 19, 2024, from https://developmentseed.org/titiler/

Tripathi, A. K., Agrawal, S. and Gupta, R. D. (2020). Cloud enabled SDI architecture: A review. *Earth Science Informatics*, *13*(2), 211–231. pp. https://doi.org/10.1007/s12145-020-00446-9

Zhao, Y., Yang, X. and Vatsavai, R. R. (2021). A scalable system for searching large-scale multi-sensor remote sensing image collections. *2021 IEEE International Conference on Big Data (Big Data)*, 3780–3783. pp. https://doi.org/10.1109/BigData52589.2021.9671679

7 Appendix

7.1 Baseline scenario questionnaire

Related to data discovery

- I am working for Wilmar in South East Asia. Do you know what is the forest baseline that I should use and where can I find it?
- I have been checking the results of the Soy map we created. Do you know which DEM was used for it? And where can I find it?
- Do you know which DEM is used as the terrain mask when using Sentinel 1 data?
- I need to access the concessions data provided by Grepalma. Where can I find it?

Related to data visualization

- I am interested on getting an overview of where was the primary forest present in Colombia in 2007. Could you visualize a layer with this data for me?
- I need to visualize the concessions provided by fedepalma. Could you do it for me?

7.2 Thematic Content Analysis prompt

```
I will give you some notes I took from an interview I did to four study subjects:
W, X, Y and Z.

Tell me if you identify any themes or topics that are repeated in the notes that
I took from the answers of the individuals. In other words, do a simple Thematic
Content Analysis of the interviews.
```

7.3 Code to evaluate request times

Disclaimer: In order to run the code presented below, the user must have authenticated their Google account and have the TiTiler-PgSTAC and the TiTiler-Xarray services running on `localhost:8082` and `localhost:8084` respectively.

```
import pandas as pd
import requests
import random

tiles = ["9/399/254", "10/800/505", "11/1603/1012",  "12/3209/2042",
"13/6407/4075", "14/12817/8159", "15/25678/16271", "16/51268/32552",
"17/102503/65134", "18/205062/130211"]

# Tiles are slightly modified to try to avoid getting cached tiles
def modify_tile(tile):
    parts = tile.split('/')
    z = int(parts[0])
    x = int(parts[1])
    y = int(parts[2])

    # Determine the range of change based on the value of z
    if z <= 9:
        change_range = 3
    elif z <= 12:
        change_range = 5
    elif z <= 15:
        change_range = 10
    elif z <= 18:
        change_range = 50

    # Apply the change to x and y
    x_change = random.randint(-change_range, change_range)
    y_change = random.randint(-change_range, change_range)

    new_x = x + x_change
    new_y = y + y_change

    # Return the modified tile as a string
    return f"{z}/{new_x}/{new_y}"

times_zarr = []
times_cog = []
z_level = []
cmap_picked = []

# The colormaps picked can be either a customized one for S11
# Forest baseline or greens_r
cmap = ["_name=greens&rescale=0,70","_name=greens_r&rescale=0,70",
```

```
        "_name=blues&rescale=0,90", "_name=blues_r&rescale=0,90",
        "_name=reds&rescale=0,80", "_name=reds_r&rescale=0,80"]

for i in range(len(cmap)):

    mod_tiles = [modify_tile(tile) for tile in tiles]
    k = i

    for tile in mod_tiles:

        url_zarr = f"http://localhost:8084/tiles/WebMercatorQuad/"+\
        "{tile}%401x?url=gs://s11-tiles/zarr/separate&"+\
        "variable=band_data&reference=false&decode_times=true&"+\
        "consolidated=true&colormap{cmap[k]}&return_mask=true"

        url_cog = f"http://localhost:8082/collections/"+\
        "Example%20FBL%20Riau/items/FBL_V5_2021_Riau_cog/tiles/"+\
        "WebMercatorQuad/{tile}%401x?bidx=1&assets=data&"+\
        "unscale=false&resampling=nearest&reproject=nearest&"+\
        "colormap{cmap[k]}&return_mask=true"

        x = requests.get(url_zarr)
        times_zarr.append(x.elapsed.total_seconds())

        x = requests.get(url_cog)
        times_cog.append(x.elapsed.total_seconds())

        z_level.append(int(tile.split('/')[0]))

        cmap_picked.append(k)

data = pd.DataFrame([cmap_picked, z_level, times_cog, times_zarr]).T
data.columns = ['colormap','zoom level','COG', 'ZARR']

data.to_csv('request_time_results_6iter.csv')
```