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

The abstract goes here.

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

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 deforestation monitoring 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).

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. Finally, 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).

Data discovery services

AMPLIAR. Maybe include other data discovery services?

A discovery service widely used for the exploration of big geo-data is Spatio-Temporal Asset Catalog (STAC). Through the standardization of spatial data, 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 that will only be read when a computation is required (Durbha et al., 2023). This feature optimizes workflows by reducing unnecessary data loading.

STAC fundamentals

A Catalog built under STAC specifications is composed by:

Table 1.1: STAC components

STAC components	Description
Assets	An asset can be any type of data with a spatial and a temporal component.

STAC components	Description
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.

Talk about the integration with sturctured data base management (PgSTAC)

Visualization interfaces

Mention also that there are multiple TiTiler services

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 dynamically generating PNG/JPEG image tiles only 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.

1.2.2 Added value of this research

expand

This research aims to develop a STAC catalog to facilitate the discovery of diverse big geospatial datasets created by different teams within the company. Depending on the time available for the thesis, data visualization functionalities may be integrated. Moreover, the research will explore tools to efficiently manage a STAC catalog containing both static and dynamic datasets.

1.3 Research questions

- What are the current challenges, practices, and user experiences related to data discovery and data visualization in the company?

- How does the integration of cloud-optimized data formats, cloud services and SpatioTemporal Asset Catalog (STAC) specifications influence the process and experiences of discovering big spatial data?
- To what extent do dynamic tiling services can perform in visualizing different cloud-optimized data formats?

2 Methodology

To address the proposed research questions, a series of activities and the deliverables expected from them are presented on Figure 2.1. Additionally, a detailed description of each activity is presented in the following subsections.

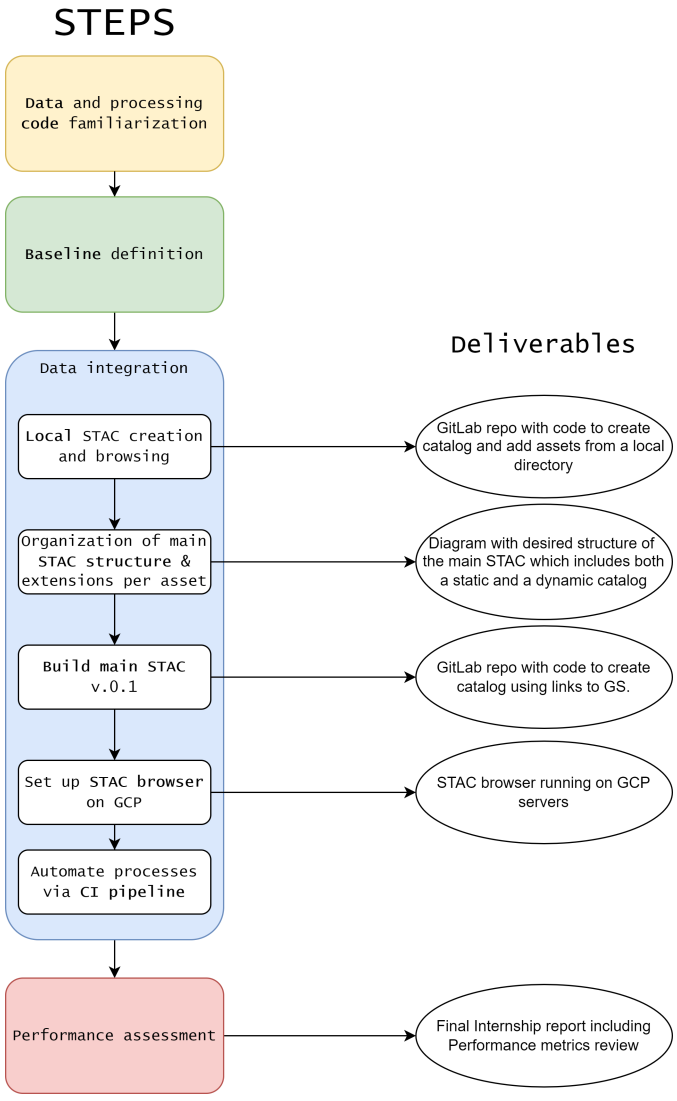

Figure 2.1: Internship flowchart

2.1 Data and processing code familiarization

In this step, I will familiarize myself with the current tools used for the processing of the images and its storage. This step will include the understanding of cloud services and internal image processing tools and the main datasets to be referenced on the catalog (STAC data). Moreover, in this step an initial description of the STAC data metadata will be performed.

2.1.1 Cloud services familiarization

An overview of the cloud services used by the company will be described. This will be mainly with the objective to understand, but not limited to:

- Who access the data?
- What are the costs of accessing it?
- How often certain data is updated?
- How is the data updated?

2.2 Baseline scenario definition

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 appendix ### and were meant to be open questions with multiple possible answers.

2.3 Data integration

2.3.1 Local STAC creation and browsing

This step will mainly be focused on the set up of the developing environment to both create a local STAC catalog and browse through it. This will include:

1. Creation of a virtual environment or Docker container with all the required packages to create a STAC catalog.
2. Creation of Local STAC using sample data from the company.
3. Browse through the STAC catalog using tools like STAC browser.

The **deliverable** of this step will be a GitLab repository with code to create a catalog, add assets from a local directory and browse through them locally using STAC browser.

2.3.2 Organization of main STAC structure & extensions per asset

In this step the structure of the STAC catalog will be defined. This will involve the selection of datasets that will be referenced on the catalog, the definition of subcatalogs and/or collections to group items with similar metadata. An initial idea of the structure of the main STAC catalog can be seen on Figure 2.2. Initially, the creation of two different subcatalogs is proposed to keep the static and dynamic dataset separated. 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.)

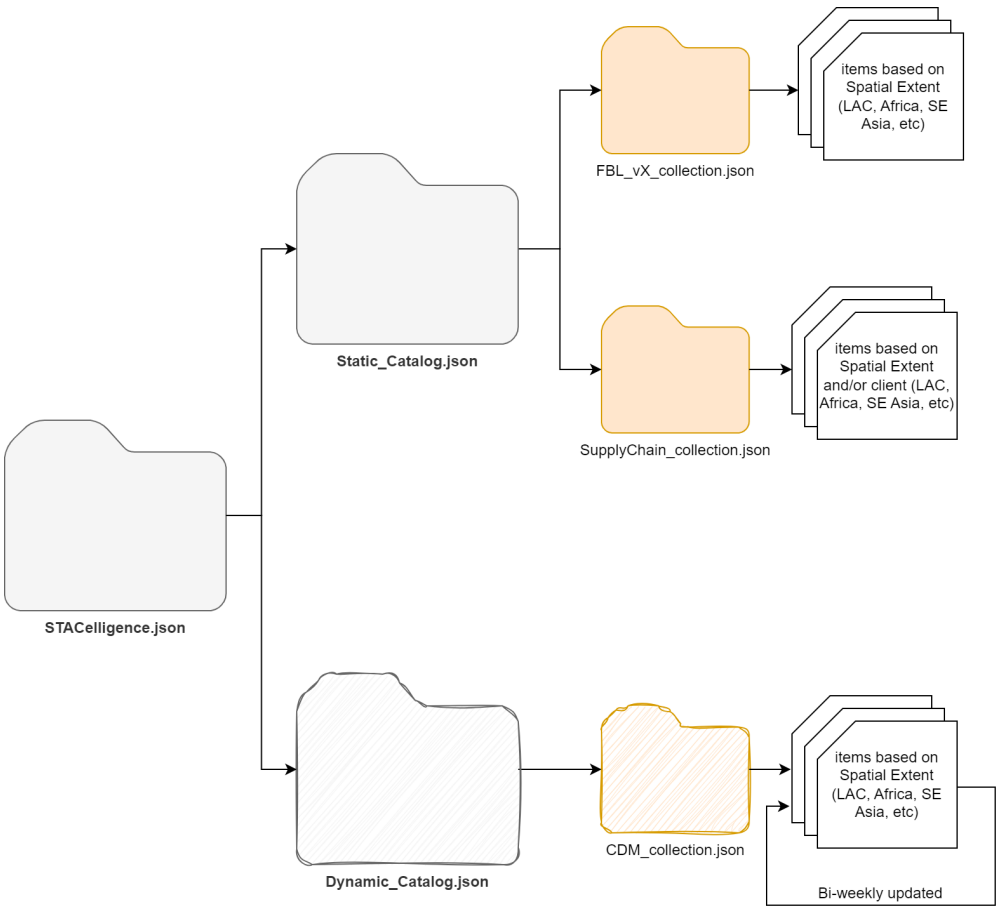

Figure 2.2: Initial proposed STAC structure

2.3.3 Build main STAC v.0.1

This step will focus on the building of the initial version of the main STAC catalog, once the datasets and the overall structure has been defined. It will involve the population of the catalog with STAC components following the defined structure on Section 2.3.2. Furthermote, on this step a series of validation tools will be used to check that the STAC catalog created is followins the STAC spcification. These tools are part of the python package stac-tools.

The **deliverable** of this step will be a GitLab repository with code to create a catalog, create collections, add assets from a directory on the cloud and update them.

2.3.4 Set up STAC browser on GCP

In this step a version of the STAC Browser application will be deployed using the tools from Google Cloud Platform (GCP). This application will allow users to browse and interact with the STAC catalog through a user-friendly interface. Additionally, this step will involve the definition of resources and tools from GCP that will be employed to deploy the application. For instance, the decision of doing it through a virtual machine or on a containerized way will be made.

The **deliverable** of this step will be a the STAC browser application running on GCP.

2.3.5 Automate processes via CI pipeline

Finally, the code to create, modify and/or deploy the STAC catalog will be merged into a continuous integration pipeline that will allow the integration of this catalog with other tools from the company. For instance, the Distributed Processing Framework (DPROF), which is satelligence's Satellite Data Processing engine.

2.4 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. 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.

The TiTiler-Xarray service was then customized to work with the specific folder structure of the ZARR overviews previously created. Dockerized versions of both TiTiler-Xarray (for Zarr files) and TiTiler-PgSTAC (for COG files) were deployed locally.

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 visualization context.

2.4.1 Speed up

The assessment of the performance of the new Data Catalog will be measured using the baseline scenario established at the beginning of the internship and the Speed Up metric proposed by (Durbha et al., 2023):

$$SpeedUp = \frac{t_{baseline}}{t_{catalog}}$$

This metric explains how much the process to access data has sped up thanks to the integration of cloud-based storage, the data catalog and the browsing interface.

3 Results

3.1 Baseline scenario

3.1.1 Main findings

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.

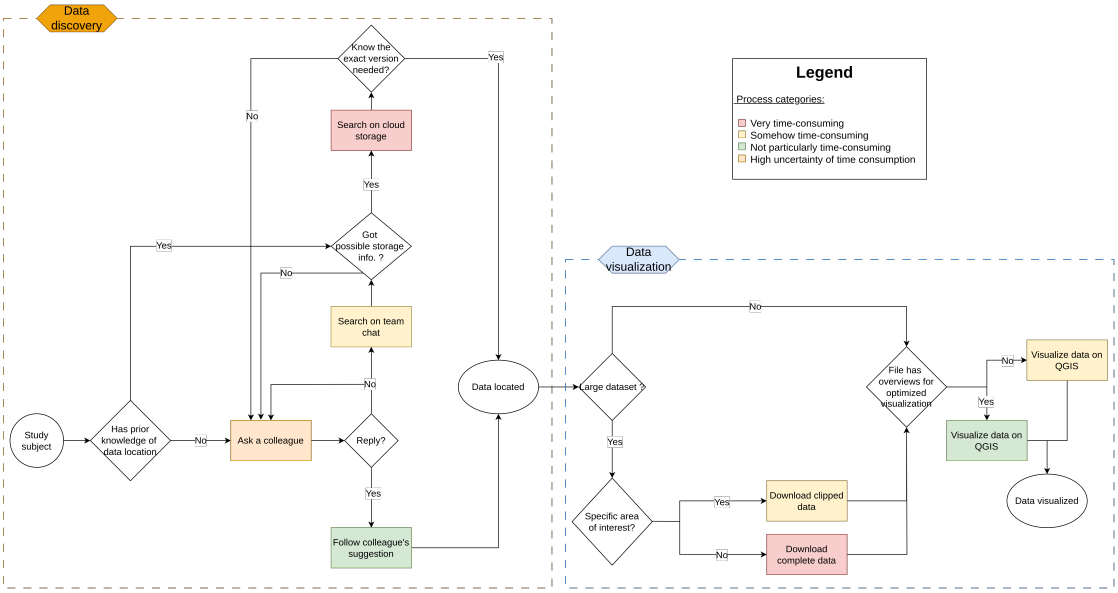

Figure 3.1: Baseline workflow

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

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.

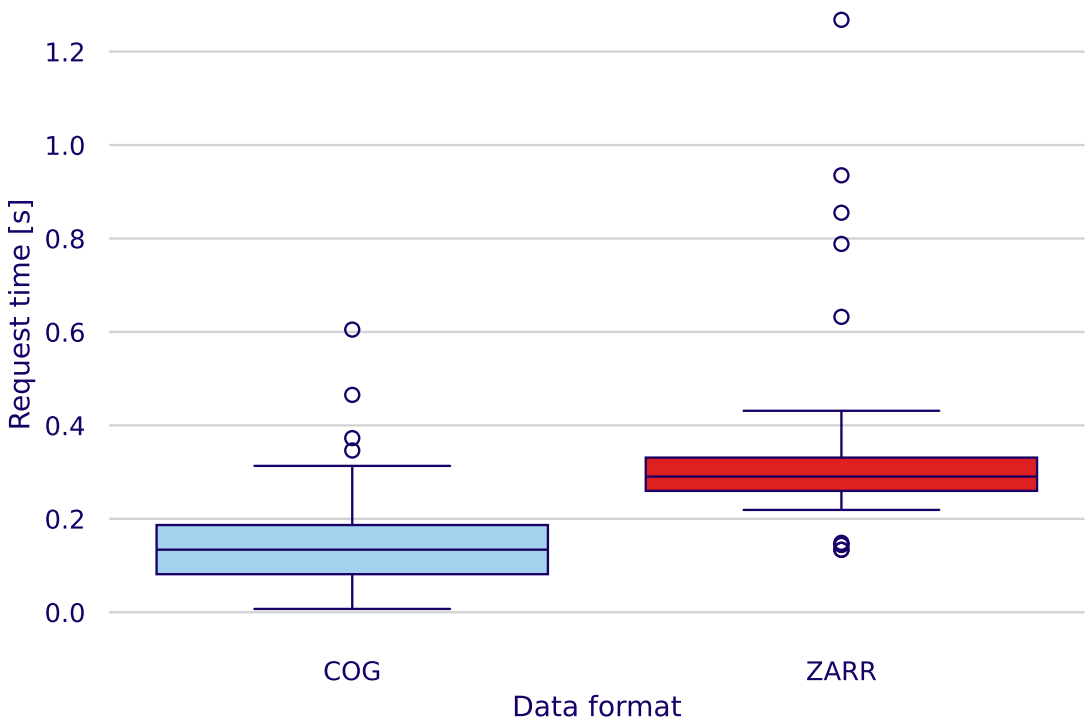

Figure 3.2: Request times depending on data format and zoom level

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.

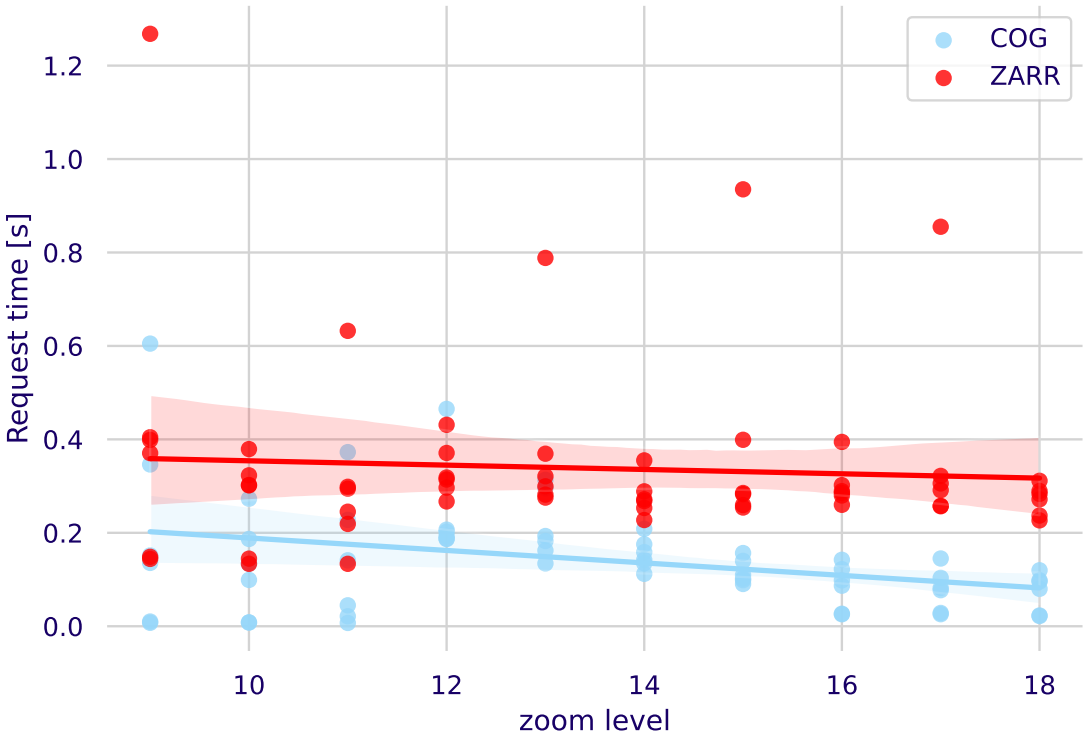

Figure 3.3: Request times depending on zoom level

4 Discussion

hgdcjfc rah fdxh sdfhfn dfayhgn d

gadfh

5 Future work

6 Internship planning

Internship duration: 08/04/2024 - 08/08/2024 (4 months)

7 References

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

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*.

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

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

8 Appendix

8.1 Code to evaluate request times

```
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://s1

        url_cog = f"http://localhost:8082/collections/Example%20FBL%20Riau/items/FBL_V5_2

        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')
```