OGC API: LDPROXY, AN UPDATED AND CONTEMPORARY APPROACH TO GEODATA PROVISION IN DISASTER MAPPING

****

****

Keywords: Natural Disasters, Open Source Geospatial Data, WebGIS, OGC API Standards, Ldproxy, Performance Comparison

Abstract

Natural disaster occurrences have been rising for the past decades. It causes severe damage both in human properties and lives. To minimize the risk caused by disaster and accelerate the recovery, the accessibility of geospatial data is critical. WebGIS utilizes the internet to make the geospatial data accessible to wider audience. In addition, the emergence of novel standards from OGC APIs also accelerates the accessibility of geospatial data. However, the abundance of Geospatial data requires a better performance of the webGIS. Ldproxy as a tool that follows OGC API standards could be an alternative to traditional tools such as Geoserver. This research aims to make a WebGIS Application for accessing geospatial data for disaster needs and to compare the performance of Ldproxy and Geoserver as a Web Map Server. The method is to build a Single Page Application (SPA) using open-source data from Google Earth Engine (GEE) and OpenStreetMap (OSM). The functionalities of the webGIS enable users to retrieve OSM and GEE data and examine the impact of the disasters. The performance of Ldproxy and Geoserver is tested using Google Light House for comparison. The parameters that are tested are Speed Index, Total Blocking Time, Time to Interactive, Max Potential First Input Delay, Network Server Latency, and Total Byte Weight. In result, Ldproxy shows slower performance in NSL compared to Geoserver. However, it offers an mvt feature format that makes the feature data can be loaded more efficiently and quickly.

1. Introduction

Natural disasters in the past five decades have shown an escalating increase in occurrence and frequency (EM-DAT, n.d.). It is also observed in 2024, that natural disaster happens in many countries. For example, in January 2024, Japan was hit by an earthquake of 7.5 magnitude. In Bandung, Indonesia, the first biggest tornado destroyed five subdistricts. In addition, floods, storms, and wildfires are the most frequently occurring disasters around the world recently. This situation caused a great loss of material and human lives.

Geographic information systems (GIS), geospatial data, and earth observation especially from remote sensing imageries play important roles especially in disaster response (Shi, et al., 2020). GIS allows analysis using available spatial data including raster and vector data for risk reduction. This includes the planning for efficient response and effective recovery efforts. Geospatial data are crucial for making more accurate and immediate decisions in case of disaster (Vučić N, 2021).

WebGIS combines GIS and geospatial data with the internet. It widens GIS and spatial data accessibility to as many users as possible in any area with an internet connection. This combination opens significant possibilities for GIS functionalities such as geospatial analysis, integration, and transmission which can improve data exchange for many sectors from local to international (Abdalla & Esmail, 2019). This will lead to the enhancement of Disaster Risk Reduction, Preparedness, and Response.

On the other hand, geospatial data is abundant. This leads to the complexity of handling and loading it on the client side. Loading geospatial data can take time as it is heterogeneous (Abdalla & Esmail, 2019). In contrast, web performance is very important and loading time is critical. The website loading time statistic proves that website performance affects user experience and user bounce rate significantly, specifically in the science field (Cai and Robert, 2023).

The presence of OGC API facilitates better integration of GIS. With specified standards of many sectors of GIS such as features, maps, styles, projection, and many more, the OGC API standards follow up acknowledged standards in information technology and allow users to communicate geospatial data seamlessly. Ldproxy, among several tools that comply with the OGC API, serves as an intermediary layer for sharing geospatial features. It has a function intersection with the well-known Geoserver as a Web Map Server. This research aims to implement an OGC API-based client-server solution to see the difference in the performance of both technologies, Ldproxy and GeoServer.

Due to its simplicity, efficiency, and performance effectiveness, a single-page application (SPA) in WebGIS for Disaster is applied in this research. In addition to these benefits, it is also very easy for the user to understand how to use it rather than a multi-page application. The application requires some functionality that makes sharing geospatial data both vector and satellite imagery for disaster management support possible. It uses both Geoserver and Ldproxy for sharing especially the feature data and the performance of both servers is tested to see which one has a better loading speed and is cost-effective. The aim is that this web application can be used by many sectors to examine the impact of geospatial data in disaster management, share it, and make accurate decisions for disaster response planning.

2. OGC API and Ldproxy

Open Geospatial Consortium APIs (OGC APIs) are a new standard that is designed to make geospatial data usage, sharing, and integration more convenient with the foundation set by the OGC Web Service Standards (including WMS, WFS, WCS, WPS, and others). These standards serve as foundational elements for crafting innovative APIs facilitating web access to geospatial data. They are specified by the criteria outlined within the OGC's Standards Program and further refined through interoperability testing within the OGC's Collaborative Solutions and Innovation Program. There are 16 API standards that currently exist. All the information above is based on https://ogcapi.ogc.org/.

Ldproxy is a service that acts as a middle layer for sharing geospatial data. It implements several OGC API standards, specifically in features, tiles, styles, routes, and 3D GeoVolumes. It is also simpler to set as Web API rather than the traditional one. Further information about ldproxy can be accessed from https://docs.ldproxy.net/de/

There is one option other than Ldproxy that follows the OGC API feature standard, it is Deegree OGC API. However, compared to Ldproxy, Deegree OGC API only implements OGC API Feature Part 1 - Core 1.0 and Part 2 Coordinate Reference System by Reference 1.0 with status implementing. While Ldproxy has more standards that have been implemented, those are:

1. OGC API - Common - Part 1: Core 1.0.0
2. OGC API - Features - Part 1: Core 1.0
3. OGC API - Features - Part 1: Core corrigendum 1.0.1
4. OGC API - Features - Part 2: Coordinate Reference Systems by Reference 1.0
5. OGC API - Features - Part 2: Coordinate Reference Systems by Reference corrigendum 1.0.1
6. OGC API - Tiles - Part 1: Core 1.0

Both Degree OGC API and Ldproxy information related to the implementation of OGC API can be accessed in https://www.ogc.org/resources/product-details/?pid=1704 and https://www.ogc.org/resources/product-details/?pid=1705.

With OGC API - Features - Part 1: Core 1.0 and OGC API - Features - Part 2: Coordinate Reference Systems by Reference 1.0 have already been official reference implementations and have higher status than implemented, which have been compliant, since August 2021. Moreover, Ldproxy already has some services that work, for instance, Daraa https://ri.ldproxy.net/daraa and Vineyards https://ri.ldproxy.net/vineyards. Ldproxy also has already a product website in GitHub form https://github.com/interactive-instruments/ldproxy. Hence, Ldproxy has been mature and therefore chosen for further research in this study.

3. Data

Two data types are used in this research, vector data and raster data. Vector data that is used here is retrieved from OpenStreetMap. While raster data that are used are satellite imagery from Google Earth Engine.

3.1 OpenStreetMap (OSM)

OpenStreetMap vector data is used in this research due to its open community-driven and open-source that covers the entire world. It is also continually updated by more than 1 million contributors allowing the data to be up-to-date. OSM has a wide catalog of data for every need. Data such as roads, buildings, land uses, rivers, lakes, and more are available there. This vector data has its type and tags to give the information about the data. There are three data types in OSM, node, way, and relation. Node is represented by one point coordinate while the way is represented by two or more points that act as a line or polygon. While a relation is the relation between the features. Furthermore, the tags consist of a key and value that describe the feature. The key describes the general category of the feature and the value is the more specific information about the feature. For example, the user can find specific information about buildings, it uses the building as a key, and the value is the more specific use of the building such as house, apartment, and many more. It is listed in the following link: https://wiki.openstreetmap.org/wiki/Map_features.

For disaster needs, OpenStreetMap offers data that is related to it. It is under specific tags of Humanitarian OSM Tags. It offers data other than data mentioned above such as utilities, public services, and physical environment, and the most related to disaster is hazard data, more specifically it offers hazard-prone areas. For more information the tags are listed in this link: https://wiki.openstreetmap.org/wiki/Humanitarian_OSM_Tags/HDM_preset.

To retrieve the data using API, overpass API is used. It functions as a web-based database that serves OSM data based on the user's query such as tags, location, and type. This also allows the developers to programmatically fetch data from an overpass in a programming language such as JavaScript.

3.2 Google Earth Engine (GEE)

For satellite imagery data that are used in this research are retrieved from GEE. It is a cloud-based platform that offers various satellite imagery and geospatial datasets. It has also an API that allows the developers to implement programs, especially those written in Python and JavaScript, to retrieve the data programmatically to the application. Presentation and analysis are performed visually on the client side by the user of the application. Remote sensing data from the optical wavelengths is therefore intended for use in the prototype developed. GEE offers three optical satellite imagery data sources: Landsat 8 and 9 which is still operating, Sentinel 2A and 2B, and MODIS. The specifications of each satellite are served in Table 1. Further detailed information can be accessed from the link below:
https://developers.google.com/earth-engine/datasets/catalog/

No	Name	Revisiting time (days)	Resolution (m)
1	Landsat 8-9	8	15,30,100
2	Sentinel 2 A-B	6	10-40
3	MODIS	1-2	250, 500, 1000, 5600

Table 1 Satellite Imageries

4. Implemented architecture

The architecture implemented here consists of three main components: client-side, server-side, and third-party data (Figure 1). Besides the components, the performance testing using Google Lighthouse is performed on the client side.

On the client side, the front end consists of CSS/Bootstrap, HTML, and JavaScript functions to make the web interface that is interactive to the user. The interaction from the user is run here where the input from the user is received and sent to the server side. Furthermore, it also handles the process of fetching data from a third party, which is OSM and GEE. It is put on the client side because the variables of the request such as date, coordinate area, satellite, and tags are also placed on the client side, so it is more efficient.

After the data is received from the client side, it is sent to the server app (NodeJS). NodeJS is programmed to send the data to Postgres as a database. The data from the database later are retrieved by Map Server which is Geoserver and Ldproxy. After that, the layer from the web map server is shown in the interface as the end product for the user.

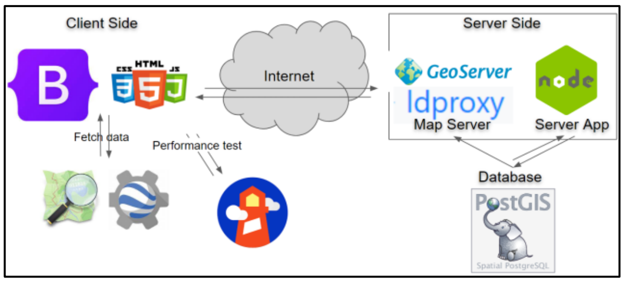


Figure 1 Web GIS implemented architecture illustration

5. Web GIS Application Functionalities

There are three main functionalities in the web application, OSM data retrieval, GEE data retrieval, and In-situ data collection. Each functionality has its input parameters from the user that should be specified. For the OSM data retrieval, the user should specify the rectangular area of interest, the type of tags (node or way), the key, and the value (optional). For the GEE retrieval, the user should specify the rectangle area of interest, the date of the remote sensing image, and the type of remote sensing images that are available Landsat, Sentinel 2, and Modis. Lastly, for the In-situ data collection, the user can see the impact of the disaster from remote sensing imageries and then delineate the area of the impact. Then The user should give the ID and the tags as notes such as the date of the disaster and the type of disaster.

6. Performance test

6.1 Testing Criteria

There are five testing criteria, those are service used, area, amount of data, and zoom level.

6.1.1 Service used: The service used by Geoserver is WFS. Unlike WMS which is very simple and still can be loaded easily due to its lightweight, WFS will be very heavy to be loaded when the data is significantly abundant. However, WMS cannot be queried based on their attribute. Therefore, WFS is still important to be used. Hence, it is proposed to use Ldproxy to see whether it can boost the loading of vector features.

6.1.2 Area: The location that was chosen for the testing is Tokyo. Based on the United Nations Department of Economic and Social Affairs in 2018, Tokyo is one of the biggest cities with the largest population cities in the world that is prone to disaster. Disasters such as tsunamis, earthquakes, and floods threaten Tokyo. Therefore, those are the reasons why Tokyo is chosen to be the area of testing in this research.

6.1.3 Amount of Data: The amount of data used here is 142.457. With this amount of data, the network performance to load the data takes a significantly slower time. Therefore, this research would like to know the quantitative measure of how much the difference between Ldproxy and Geoserver is to load the data.

6.1.4 Zoom Level: Two zoom levels are used in this research (Figure 29). Zoom level 13 is used because it shows the whole feature while Zoom level 15 is used because it shows only parts of the feature.

6.1.5 Test Quantity: The amount of data acquisition in each service for every zoom level and every parameter is 20. For example, ldproxy in Zoom level 13 for Speed Index is tested 20 times. The reason is to see the real pattern and avoid outlier values due to internet connection instability.

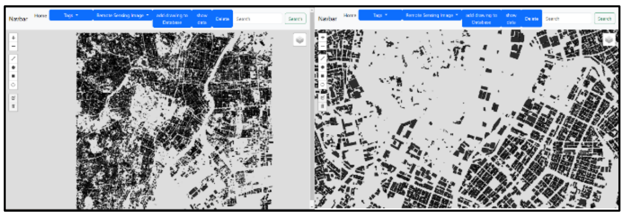


Figure 2 data visual from Zoom level 13 (left) and Zoom level 15 (right)

6.2 Testing Parameters

The tool used for testing the performance is Google Lighthouse. The performance of web pages is measured by six main parameters, those are First Contentful Paint (FCP), Speed Index (SI), Largest Contentful Paint (LCP), Total Blocking Time (TBT), Cumulative Layout Shift (CLS) (Table 2). However, FCP, LCP, and CLS are not directly affected by Geoserver or Ldproxy. Only the other three that is affected by Geoserver and Ldproxy. Furthermore, there are additional parameters that are affected by the Geoserver or Ldproxy but are not directly weighted or have a minimum effect on the performance of the web pages, those are Max Potential First Input Delay, Network Server Latency, and Total Byte Weight. All the details and explanations here can be accessed and learned from the Google Lighthouse website: https://developer.chrome.com/docs/lighthouse/performance/performance-scoring.

Audit	Weight (%)
First Contentful Paint	10
Speed Index	10
Largest Contentful Paint	25
Total Blocking Time	30
Cumulative Layout Shift	25

Table 2 Main Parameters of Web Page Performance

6.2.1 Speed Index (SI): The speed index measures how much time it takes for the content of the page to be visibly populated. The unit is in second. This is the main measurement that is weighted directly to the performance of the web page. It classifies the web page into three categories, fast, moderate, and slow (Table 3).

Speed Index (seconds)	Class
0-3.4	Fast
3.4-5.8	Moderate
>5.8	Slow

Table 3 Speed Index Classes

6.2.2 Total Blocking Time (TBT): Total Blocking time measures the sum of all periods between the First Contentful Paint (FCP) or the first content that appears on the page becomes interactive. It is the time when the web page is blocked from user input such as typing and mouse click. It begins to measure when it has already exceeded 50 milliseconds. For example, if the time it takes from FCP to interactive takes 120 milliseconds, the measurement will be 70 milliseconds. TBT is the parameter that is weighted for the performance. It is divided also into three classes (Table 4).

TBT (milliseconds)	Category
0-200	Fast
200-600	Moderate
>600	Slow

Table 4 Total Blocking Time Classes

6.2.3 Time to Interactive (TI): Time to Interactive measures the amount of time it takes for the page to become fully interactive. There are three classes that state that the less time it takes the better the performance (Table 5). TTI is also one of the parameters that are weighted directly for the performance of the web page.

TI (seconds)	Category
0-3.8	Fast
3.9-7.3	Moderate
>7.3	Slow

Table 5 Time to Interactive classes

6.2.4 Max Potential First Input Delay (FID): First Input Delay measures the time needed from when the user interacts with the web page, for example by clicking a link or button, to the time when the browser begins to process. Therefore, max FID is the time for the longest FID. The classes in FID are shown in Table 6.

FID (milliseconds)	Category
0-130	Fast
130-250	Moderate
>250	Slow

Table 6 FID Classes

6.2.5 Network Server Latency (NSL): Network Server Latency measures the time needed by the server to respond and process the request. There is no class for NSL. The measurement unit is in milliseconds. The higher value of NSL indicates the poorer the server works.

6.2.6 Total Byte Weight (TBW): Total byte weight is the total amount of data loaded on the web pages. From the web page point of view, it affects the time to load the feature to web pages. The more data it used the longer it takes to be loaded. It is important also from the user's point of view because the more weight it affects directly the money it costs from the user. It is measured in kibibyte or kilo binary bytes.

7. Results and discussion

7.1 Web-GIS Application

The application is equipped with three main functionalities: vector data from OpenStreetMap retrieval, remote sensing data from Google Earth Engine retrieval, and in-situ data drawing. The idea is for the user to be able to mark the impact caused by the disaster in the remote sensing images before and after the disaster takes place and analyze what and how many features, in the form of vector data, were affected.

7.1.1 OpenStreetMap Data Retrieval: In this functionality, the user needs to specify the area (rectangle), the tag type (node or way), and the key and value (optional) (Figure 3). The OSM tags list is provided in the OSM TAGS hyperlink text in the Tags dropdown button that connected to https://wiki.openstreetmap.org/wiki/Map features (Figure 4).

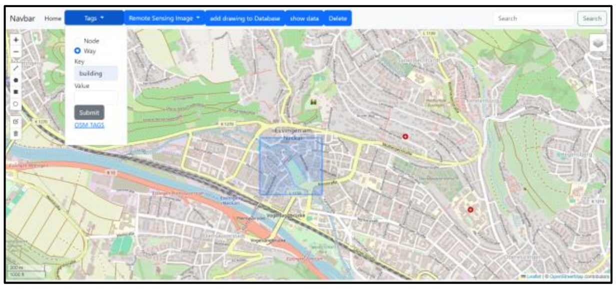
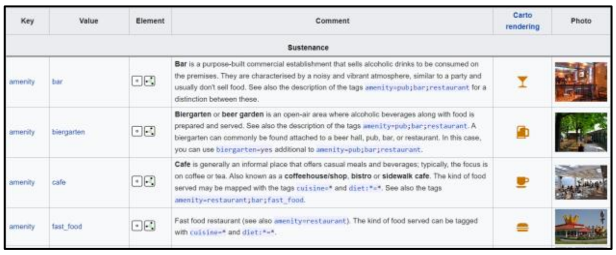


Figure 3 OSM data retrieval.



Key	Value	Element	Comment	Carto rendering	Photo
			Sustenance		
amenity	bar		**Bar** is a purpose-built commercial establishment that sells alcoholic drinks to be consumed on the premises. They are characterised by a noisy and vibrant atmosphere, similar to a party and usually don't sell food. See also the description of the tags amenity=pub;bar;restaurant for a distinction between these.		
amenity	biergarten		**Biergarten** or **beer garden** is an open-air area where alcoholic beverages along with food is prepared and served. See also the description of the tags amenity=pub;bar;restaurant. A biergarten can commonly be found attached to a beer hall, pub, bar, or restaurant. In this case, you can use biergarten=yes additional to amenity=pub;bar;restaurant.		
amenity	cafe		**Cafe** is generally an informal place that offers casual meals and beverages; typically, the focus is on coffee or tea. Also known as a **coffeehouse/shop**, **bistro** or **sidewalk cafe**. The kind of food served may be mapped with the tags cuisine=* and diet:*=*. See also the tags amenity=restaurant;bar;fast_food.		
amenity	fast_food		Fast food restaurant (see also amenity=restaurant). The kind of food served can be tagged with cuisine=* and diet:*=*.		

Figure 4 OSM Tags list example

Here is what it looks like as an example when the user specifies the area, the tag as way, and the key as building. As shown in Figure 5, all the buildings are retrieved and saved to the database and shown by the Geoserver or Ldproxy to the user.

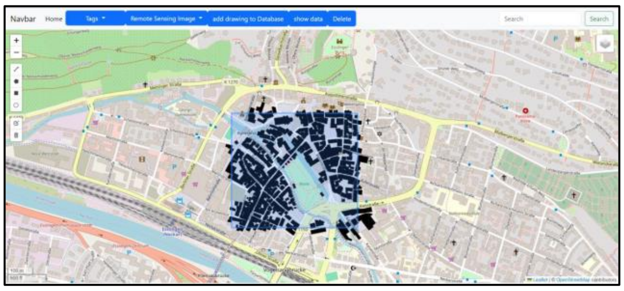


Figure 5 Result of OSM data retrieval

7.1.2 Google Earth Engine Data Retrieval: For the GEE data retrieval, the user needs to specify the area, date, and remote sensing mode that wants to be used. The options available now are Landsat 9, Sentinel 2, and Modis. Here as an example in Figure 6, the user needs to specify the Remote sensing data as Landsat 9 and Sentinel 2, the area in Mount Bromo in Indonesia, and the date is 16th September 2023. There was a forest fire in that area around 6th September 2023.

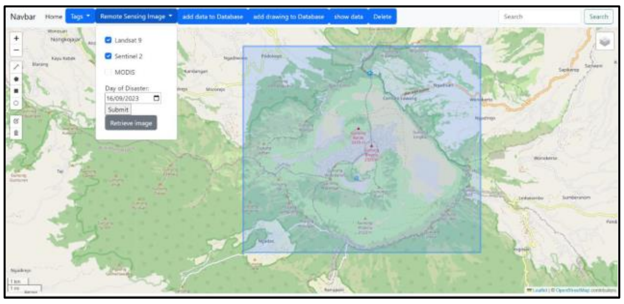


Figure 6 Input for GEE data retrieval

As a result, in Figure 7, the satellite images will be shown as a pair, before (top) and after (bottom). The "before" image is an image between the date specified and 3 months back with the lowest number of clouds. The "after" image is the image available to the nearest date specified. It can be seen that there were burning marks in that area. Nevertheless, occasionally the image is covered by clouds making it hard to examine what is happening on the surface.

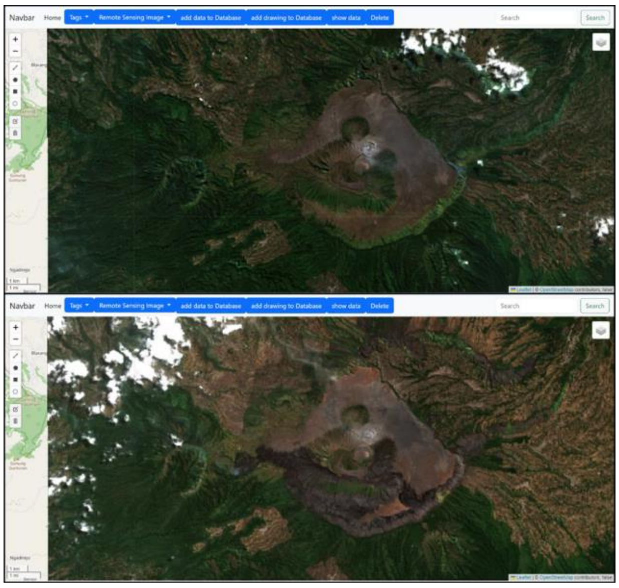


Figure 7 Result of GEE data retrieval

7.1.3 In-situ Data Collection: Continuing the functionality before, the burning marks in the example above can be digitized using this functionality. Users can draw polygons following the impact of the disaster visible from the satellite images (Figure 8).

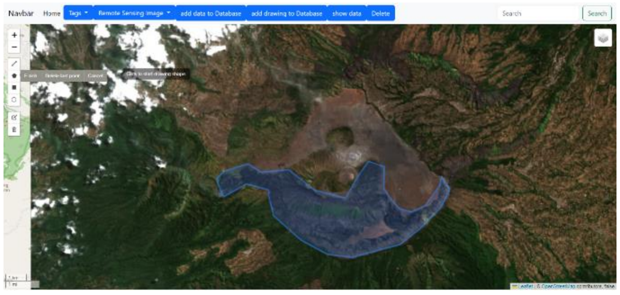


Figure 8 user defines area

Then after finished digitizing, the polygon should be clicked and it will pop up the input window for ID and tags (Figure 9).

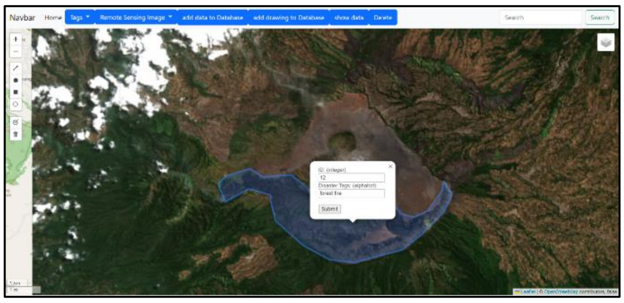

ID: (integer)
12
Disaster Tags: (alphabet)
forest fire

Figure 9 Pop-up input field for the ID and tags.

The user should specify the ID and tags. Then the user can save the data by clicking the add drawing to database button in the menu bar on top of the window. The data will be shown after clicking the show data button (Figure 10).

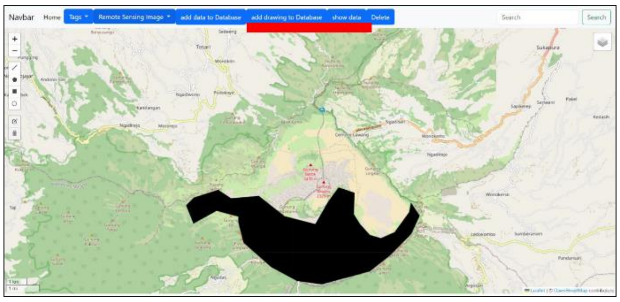
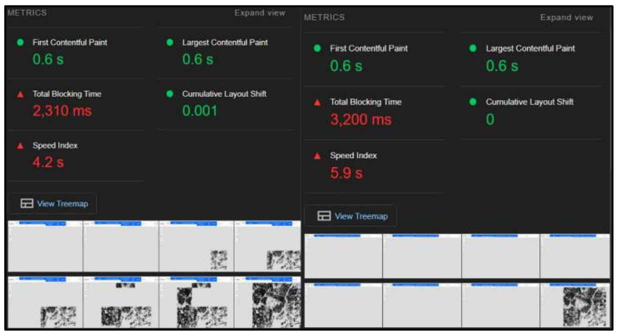


Figure 10 Result of In-situ data Collection functionality

7.2 Performance Test

The performance test is conducted in the Chrome browser using the Google Lighthouse plug-in. Here is what it looks like for the interface of the performance testing (Figure 11). It is shown in the bottom of the figure, that ldproxy on the left side loads the


METRICS

First Contentful Paint 0.6 s
Largest Contentful Paint 0.6 s
Total Blocking Time 2,310 ms
Cumulative Layout Shift 0.001
Speed Index 4.2 s

METRICS

First Contentful Paint 0.6 s
Largest Contentful Paint 0.6 s
Total Blocking Time 3,200 ms
Cumulative Layout Shift 0
Speed Index 5.9 s

Figure 11 Performance test interface for Ldproxy (left) and Geoserver (right)

Shown below are the performance test graphs for every parameter. From the left to right column in the x-axis are Geoserver Zoom level 13, Geoserver Zoom level 15, Ldproxy Zoom level 13, and Ldproxy Zoom level 15. The blue dot in the middle of the population represents the median value of each column while the red dot represents the mean value (Figure 12, Figure 13, and Figure 14).

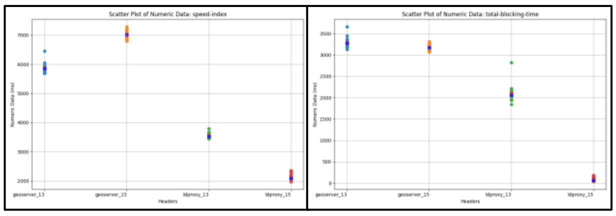


Figure 12 Performance test graphs for Speed Index (left) and Total Blocking Time (right)

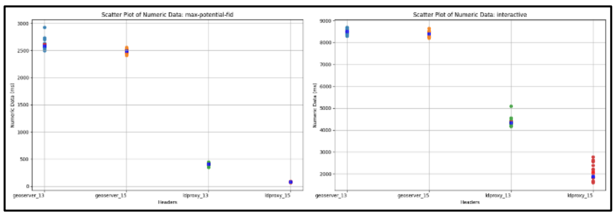


Figure 13 Performance test graphs for Max Potential FID (left) and Time to Interactive (right)

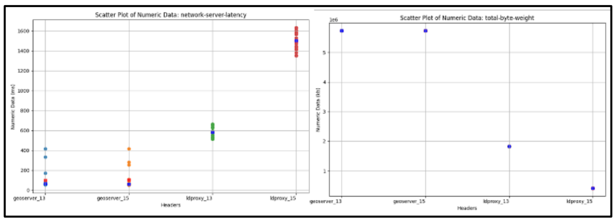


Figure 14 Performance test graphs for Network Server Latency (left) and Total Byte Weight (right)

	Geoserver		Ldproxy	
Zoom Level	13	15	13	15
SI (s)	5.8	7.0	**3.5**	**2.1**
TBT (s)	3.3	3.2	**2.0**	**0.1**
TI (s)	8.5	8.4	**4.3**	**1.9**
FID (s)	2.6	2.5	**0.4**	**0.08**
NSL (s)	**0.06**	**0.06**	0.6	1.5
TBW (Mib)	5597.2	5597.2	**1777.9**	**402.5**

The performance test's result is summarized in
. with the measurement units (s) as second and (Mib) as Mebibyte. The number served here is from the median value to avoid the effect of the outlier. The bolded number is to mark the better value compared to the other service

It is clear that Ldproxy is better than Geoserver except in NSL. This means that the time to process the request and send it back to the client in Ldproxy takes more time than Geoserver. However, because the mvt file format that is used in Ldproxy, enables the data to be gridded and loaded as lighter tiles, it results in faster loading and processing time. It is also more efficient as it is stated in the TBW which the data is much less heavy.

	Geoserver		Ldproxy	
Zoom Level	13	15	13	15
SI (s)	5.8	7.0	**3.5**	**2.1**
TBT (s)	3.3	3.2	**2.0**	**0.1**
TI (s)	8.5	8.4	**4.3**	**1.9**
FID (s)	2.6	2.5	**0.4**	**0.08**
NSL (s)	**0.06**	**0.06**	0.6	1.5
TBW (Mib)	5597.2	5597.2	**1777.9**	**402.5**

Table 7 Performance test result

From the results, it is safe to conclude that Ldproxy is superior to Geoserver. Ldproxy outperformed Geoserver in almost all testing parameters except Network Server Latency. This means that Ldproxy takes more time to deliver the request, process, and send the result back to the user. However, another variable that is offered by both tools needs to be considered. The feature format offered by Ldproxy is mvt, which slices the overall features into tiles and shows the features that the location lies within the screen view at a certain zoom level. While Geosever uses traditional WFS. This enables Ldproxy to have more lightweight data compared to the WFS of Geoserver. As a result, Ldproxy has better performance in the other five testing parameters and it compensates for the long latency time.

8. Conclusion

This research advocates the utilization of open-source data and applications for disaster necessity. The data used are from vector data from OpenStreetMap and remote sensing images from Google Earth Engine. The challenges lie in ensuring that the data are accessible, how the application has functionality that can be utilized for disaster-related requirements, and guaranteeing that the application has cutting-edge performance. The methods are to establish webGIS for disaster needs and to implement OGC API conformant Ldproxy as the Web Map Server and compare it to traditional WFS technology from Geoserver.

Single Page Application webGIS is established with several functionalities that allow the user to retrieve the data and observe the impact of the disaster. The satellite imagery retrieval has a date, area, and type filter and the data retrieved includes the data with a specific date specified by the user and the images before the disaster happened. This Facilitates the user to observe the visual difference before and after a disaster and define the disaster coverage. The application also is equipped with the ability to draw polygons to delineate the disaster coverage and save it to the database. The vector data from OSM enables the user to list features, such as buildings, roads, land, etc. that are affected by the disaster. Hence, with these data and functionalities, the application provides sufficient features for disaster requirements, especially in the disaster response stage.

Geoserver has been the most used and powerful Web Map Server to publish spatial data. However, the limitation of data format choice and performance could be the issue to overcome the abundance of spatial data. The emergence of OGC API conformant Ldproxy is proposed to challenge Geoserver as a tool that publishes spatial features.

The Performance of both Geoserver and Ldproxy is tested in the application front-end using Google Lighthouse. The tests were conducted with two different zoom levels, zoom levels 13 and 15 with 142.457 features. Six parameters are used to measure the performance of both tools. Those parameters are Speed Index, Total Blocking Time, Time To Interactive, Max Potential First Input Delay, Network Server Latency, And Total Byte Weight. Each zoom level in one tool for every parameter has been tested 20 times.

The results are that Ldproxy outperformed Geoserver in almost all parameters except network server latency. The network server latency of Ldproxy is 10 and 25 times higher than Geoserver in Zoom levels 13 and 15 respectively. This means that it takes more time to process the request and send the result back to the user. In other words, the performance of the server itself is poorer. However, Ldproxy offers more feature formats, one of which is mvt. This mvt feature format slices the vector data into tiles resulting in lighter data weight and requiring less time to load. Specifically, in zoom level 15, where the data is not all loaded, Ldproxy can efficiently display the data within the active screen. Consequently, Ldproxy has better performance in the rest five testing parameters. In conclusion, Ldproxy is still a better choice despite poorer performance thanks to the availability of the mvt feature format.

References

Abdalla, R., & Esmail, M. (2019). *WebGIS for disaster management and emergency response.* Cham: Springer International Publishing. doi:10.1007/978-3-030-03828-1

Cai and Robert. (2023, August 1). *Website Loading Time Statistics (2023)*. Retrieved February 16, 2024, from Tooltester: https://www.tooltester.com/en/blog/website-loading-time-statistics/

EM-DAT. (n.d.). *The International disaster Database*. Retrieved September 23, 2023, from https://www.emdat.be/

Shi, P., Du, J., Xu, W., Zhou, T., Wang, Y., Ye, T., . . . Jaeger, C. (2020). Disaster Risk Science: A Geographical Perspective and a Research. *International journal of disaster risk science, 11*(4), 426-440. doi:https://doi.org/10.1007/s13753-020-00296-5

Vučić N, C. V. (2021). Importance of Official Geodata in Disaster Risk Management—Case Study of Croatia. *Earth, 2(4)*, 943-959. doi:https://doi.org/10.3390/earth2040055