LEGACY VS MODERN: A COMPREHENSIVE STUDY AND COMPARATIVE ANALYSIS OF OGC STANDARDS FOR PUBLISHING GEOSPATIAL DATA

Nikolay Krastanov

M. Eng., Nikolay Krastanov
Ellatzite-Med AD, Mirkovo, Sofia District, Bulgaria
E-mail: nikolai.krastanov@gmail.com

Abstract
Nowadays GIS (Geographic Information System) web applications play an emerging role in different life areas. They are not just a base map with pinned objects on various locations. They are a complex set of tools for analysis of different natural phenomena, disasters, trends, infrastructure, etc. Usually a GIS web application consists of a three layer structure – a data layer which is the core of the application, a layer responsible for data publication and a frontend for data visualization. This study is focused on comparing two OGC (Open Geospatial Consortium) standards for data publication – OGC APIs (Application Programing Interface) and OGC Services. In other words this paper compares legacy approach for publishing geospatial data with a modern one. For the test scenario open source technologies are used. Data layer is based on PostgreSQL (PostGIS), data publication functionality is provided by Geoserver and QGIS server which implement the legacy standard and pygeoapi and ldproxy implementing the contemporary approach. OpenLayers library is used to serve the frontend.

Keywords: *geospatial data, spatial data infrastructure, API, OGC, Geosserver, QGIS Server, pygeoapi, ldproxy*

INTRODUCTION

Web based GIS (Geographic Information System) solutions are getting more and more widespread and are facilitating different life areas such as industry, urban development and planning, environmental activities, climate changes, etc. [1]

The most important part of every application is data which it operates on. In the world of GIS, in order to make data reach the end user – to get it from the DBMS (Database Management System) and to visualize it in the GUI (Graphic User Interface) an intermediary is needed. Its role is to query the DBMS for data according to some criteria and to publish it as per some rules, policies and standards.

The aim of this research is to compare some of the most popular open source products for publishing geospatial data.

In Ref. [2] authors compare Geoserver and pygeoapi using ALKIS data. Aspects such as installation process, folder structure, layer preview, rendering times against numerous features and zoom levels are evaluated in order to compare both implementations.

Similar research by Sulaemanand and Behr [3] is made comparing Geoserver and ldproxy. The parameters they evaluate are similar to those in [2], focusing on these which can be measured quantitatively such as First Contentful Paint (FCP), Speed Index (SI), Largest Contentful Paint (LCP), Total Blocking Time (TBT), Cumulative Layout Shift (CLS).

Authors of Ref. [4] focus on performance and compare OGC API and OGC Web Feature Service in terms of network latency mainly.

This paper is also focused on comparing the solutions mentioned above (Geoserver, pygeoapi and ldproxy) and QGIS server in addition.

To be more precise, it is rather a comparison of standards [5] for publishing geospatial data.

From chronological point of view, first approach is based on OGC Web Services (OWS) which relies on SOAP (Simple Object Access Protocol) and XML (eXtensible Markup Language), while the newer one implements OGC API

(Application Programming Interface) and uses JSON (JavaScript Object Notation). The later one follows the principles of modern web application.

Evaluated parameters in this research are similar to those in Ref. [2] and Ref. [3], but the focus is on performance (CPU load, used memory, rendering times) and integration with other systems and solutions.

METHODS

The methodology used in this analysis consists of three main points:

- System architecture
- OGC standards.
- Analysis.

System architecture

Dataset

The test scenario for the study is based on the following solution. Firstly, DBMS which serves the data is based on PostgreSQL with PostGIS extension. Database contains geospatial data of big mining company (mainly infrastructure and cadastal data) with a total number of 45 feature classes and around 10000 records.

Geospatial layer

The second layer, which is the real focus of the research can be presented by two groups of software products. On the one hand are those which implement OGC Web Services standards – Geoserver and QGIS server and on the other pygeoapi and ldproxy which are OGC API compliant. To evaluate them under equal conditions a containerization approach is chosen based on Docker [6].

The concept behind containerization is not going to be discussed as it is not the aim of this paper. Along with the advantages that this approach provides such as isolation, fast recovery, etc. in this particular case it facilitates measurement of consumed resources – CPU load, used memory, disk usage.

Web GIS application

As a third layer, a simple SPA (Single Page Application) is designed, based on OpenLayer JavaScript library. It just visualizes published geospatial data.

The architecture is shown in Figure 1.

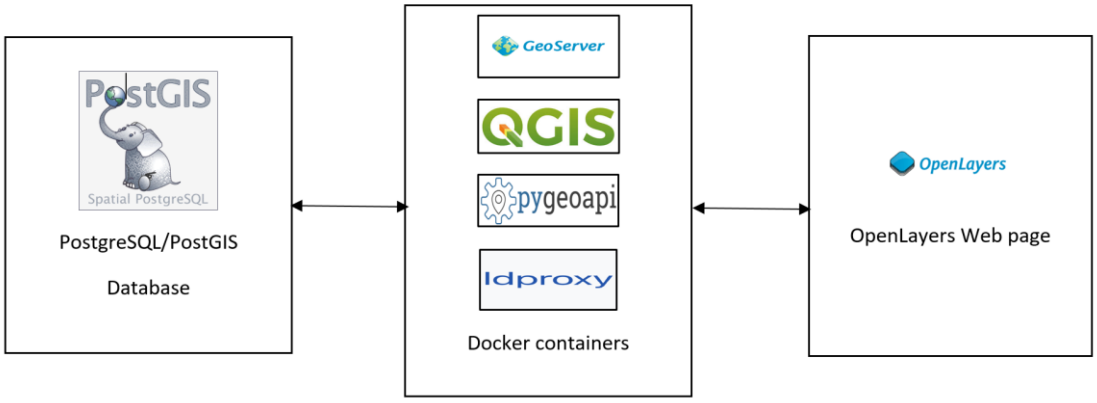


Figure 1. System architecture

OGC standards

As it was mentioned above it is rather a comparison between two approaches for publishing geospatial data, but to be even more precise it is a comparison between to data representations – XML and JSON. Both are used to exchange data between different platforms and applications.

The initial version of XML was released in 1998 by the XML working group as part of the Standard Generalized Markup Language. It is a text format which represents data in tree like structure.

JSON was released in 2001 by Douglas Crockford and Chip Morningstar and is derived from JavaScript. Again, it is a text format that uses “key -> value” pattern to create map-like structure.

Detailed comparison of both formats is given in Ref. [7]. Just to illustrate how data is represented in both formats a sample data structure is shown in the next two figures (Figure 2 and Figure 3) – in XML and JSON format, respectively.

```
<feature>
  <attribute1>Attribute 1 Value</attribute1>
  <attribute2>Attribute 2 Value</attribute2>
</feature>
```

Figure 2. XML encoded data structure

```
{
  “attribute1” : “Attribute 1 Value”,
  “attribute2” : “Attribute 2 Value”,
}
```

Figure 3. JSON encoded data structure

As figures show – the same data structure is more compact and readable when presented in JSON. This means that in terms of network traffic a data structure encoded in JSON will generate smaller payload than encoded in XML.

In addition to what has been said about advantages of containerization it must be complemented that thanks to this, the ability to setup just one test system for observation has been achieved. The only thing that need to be done is to switch between URLs (Uniform Resource Locator) of provided services (containers). If monolithic concept is chosen, it would practically lead to complex solution (actually four) as all four software products have specific installation process.

OGC Web Services

OGC Web Services are set of standards that strictly and clearly define a way of communication between client and geospatial server using XML and HTTP (Hypertext Transfer Protocol). They provide a vendor-neutral, interoperable framework for web-based discovery, access, integration, analysis, exploitation and visualization of multiple online geodata sources, sensor-derived information, and geoprocessing capabilities. [8] OWS include protocols such as WFS (Web Feature Service), WMS (Web Map Service), WCS (Web Coverage Service), WPS (Web Processing Service) and WMTS (Web Map Tile Service).

OGC API

OGC API is a modern set of standards which aims to replace OWS protocols. Standards are resource oriented and use RESTful (Representational State Transfer) architecture, JSON as data format and HTTP as transport protocol. They are

modular by design which allows only necessary modules to be included. OGC API includes OGC API - Features, OGC API - Tiles, OGC API – Maps, etc.

RESULTS

Programming languages

In order to understand the nature of the evaluated software solutions, it is good to say a few words about the programming languages which the applications are implemented on.

Pygeoapi is a Python application. Python is relatively new and modern language. In the common case, in order to execute a Python program, a Python interpreter is needed which affects the performance.

Ldproxy and Geoserver are a Java applications which means that a JVM (Java Virtual Machine) is needed. Thus an intermediary is needed again which also affects the performance. JVM allows precise fine-tuning and in this way minimization of performance issues can be achieved.

QGIS server is native application written in C++ which presumably means that under equal conditions will show best performance compared to others.

Installation process

For the purposes of this study, docker images of the four solutions are used. Images are available at DockerHub. This simplifies the process of installation pretty much as the only things to do are: download images and run a docker command for starting a particular container. Within seconds all is up and running. Thus it is appropriate for testing purposes.

The monolithic concepts suggests having specific knowledge on operating systems, installing dependencies, JVM (Java Virtual Machine), etc.

Configuration

Two of the products pygeoapi and ldproxy uses YAML (YAML Ain't Markup Language) as file format for their configuration files. Despite that format is human-readable, it is indentation-based which sometimes may lead to confusion. Another disadvantage of the configuration of these two solutions is that everything that is aimed to be published, must be included in the configuration files. There are tools that facilities this process, but they require additional configuration and some of them raise concerns about security.

Geoserver keeps its configuration in XML and Java properties files in its data folder. It might be said that they are also human-readable but knowledge of XML syntax rules is required. Because of this, Geoserver provides graphical administration interface which simplifies configuration process.

QGIS server is configured via INI (Initialization) file format or environmental variables. INI syntax is “key → value” format so it is human-readable and relatively easy to configure the server. There is a slight, but significant deference between QGIS server and the others. While they are typical server applications, QGIS is not. Its a binary program that implements CGI (Common Gateway Interface) and this way it needs a web server to work (Apache or Nginx are typical ones).

Functionality

If a complete GIS solution has to be pointed out among the four applications, with no doubt it is Geoserver. Along with OWS functionality, it provides complete graphical interface for administration as well as REST interface to interact with other applications. It also has built-in support for ActiveDirectory/LDAP integration and supports variety of authentication mechanisms which makes it suitable for enterprise usage.

QGIS server lacks administation interface, but with third party applications such as Lizmap or QWC2 a complete GIS solution might be achieved.

Two tables (Table 1 and Table 2) are presented. They are taken from QGIS server official documentation [9] and show comparison of features and standards between QGIS server and other mapping servers. Tables are up to 1st of January 2021.

Table 1. Comparison of QGIS server to other mapping server

Features	**QGIS Server**	**GeoServer**	**ArcGIS Server***
Since	2006	2001	1999
Licence	GPL	GPL	commercial
Commercial support	Multiple companies	Multiple companies	ESRI and its vendors network
Technology	C++/python	Java	C++
Tile cache	yes	yes (via GeoWebCache)	yes
3D	No	No	Yes
Querying	FES (2.0) and OGC (1.0) filters	CQL and OGC filters	OGC filters
Report generation	yes	yes	yes
Server administration	yes via third parties (LizMap, QWC2, etc.)	web + API REST	web + API REST
GIS project Layer/symbology edition	complete via dedicated GUI	simple via web interface	complete via dedicated GUI

Table 2. Comparison of OGC specification versions implemented in QGIS server and other mapping servers.

OGC standards	**QGIS Server**	**GeoServer**	**ArcGIS Server***
WMS (Web Map Service)	1.3.0 - 1.1.1	1.3.0 - 1.1.1	1.3.0 - 1.1.1
WFS (Web Feature Service)	1.1.0 - 1.0.0	2.0.0 - 1.0.0	2.0.0 - 1.0.0
OAPIF (aka WFS3)	1.0.0	no	no
WMTS (Web Map Tile Service)	1.0.0	1.0.0	1.0.0
WCS (Web Coverage Service)	1.0.0	2.0.1 - 1.0.0	2.0.1 - 1.0.0
WPS (Web Processing Service)	no	1.0.0	1.0.0
CSW (Catalogue Service for the Web)	no	2.0.2	no
SLD (Styled Layer Descriptor)	yes	yes	yes

**ArcGIS is proprietary solution and is not part of this research.*

On the next table (Table 3) a comparison between ldproxy and pygeoapi is given. They are compared against OGC API standards. Sources of information in Table 3 are: Ref. [10] and Ref. [11].

Table 3. Comparison between OGC API implementations in ldproxy and pygeoapi

OGC API	ldproxy	pygeoapi
Features Part 1: Core	compliant	compliant
Features Part 2: Coordinate Reference Systems by Reference	compliant	compliant
Features Part 3: Filtering	compliant	
Features Part 4: CRUD	compliant	
Features Part 5: Schemas	compliant	
Tiles Part 1: Core	compliant	
Routes Part 1: Core	compliant	
3D Geo Volumes	compliant	
Common Query language (CQL 2)	compliant	
Environmental Data Retrieval Standard		compliant
Maps Part 1: Core		implementing
Processes Part 1: Core		compliant
Records Part 1: Core		implementing

Performance analysis

For the test case a virtual machine with Ubuntu Server 22.04 LTS is used. Allocated resources are: 4 core CPU, 16 GB of RAM. All components are located on this machine in order to avoid network latency.

As it was mentioned the focus of this research will be a performance evaluation on the four implementations. The chosen test case is OGC Services WFS against OGC API Features as aim of both is provide equal functionality. A multipolygon layer is chosen with 1683 features and 34 attributes for each feature. Tests are repeated 20 times to increase precision and to avoid random results. Average values are presented.

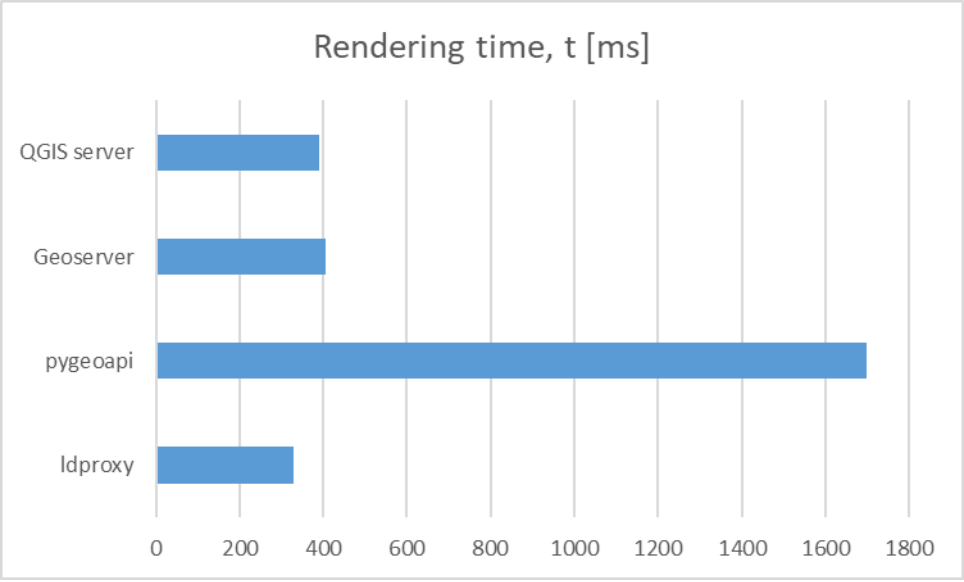


Figure 4. Rendering time of QGIS server, Geoserver, pygeoapi and ldproxy.

As seen of Figure 4, rendering times are almost identical except for pygeoapi which is significantly slower than others. Ldproxy has best result with 334 ms, followed by QGIS server with 393 ms and Geoserver with 407 ms. One of the

possible reasons that native application (QGIS) is outperformed by Java application (ldproxy) might be heavier XML parsing process.

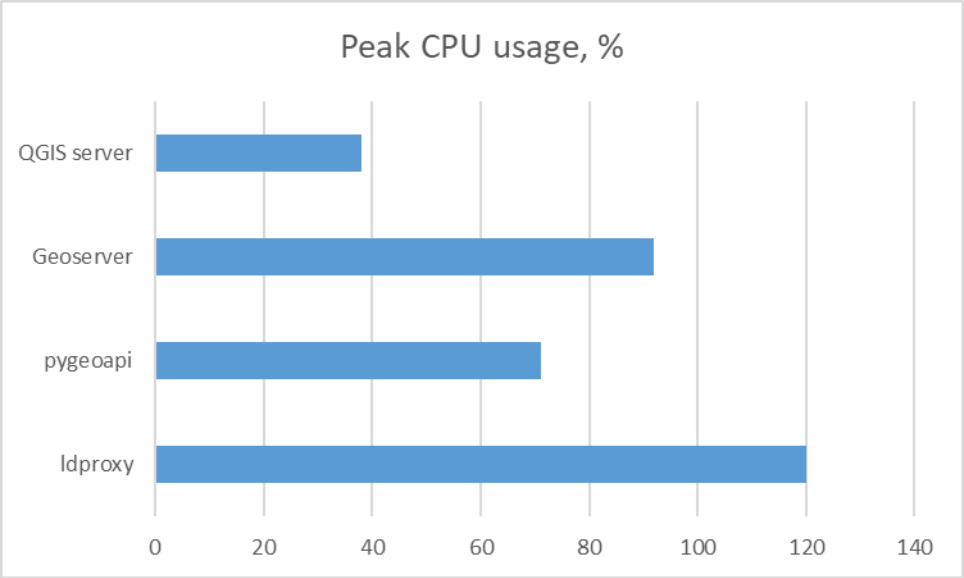


Figure 5. Peak CPU usage of QGIS server, Geoserver, pygeoapi and ldproxy.

As it comes to peak CPU usage, QGIS outperforms the rests with 38% which is expected as it is native application (Figure 5). It is followed by pygeoapi with 71%, Geoserver with 92% and ldproxy with 120%. This is a peak usage during serving a request.

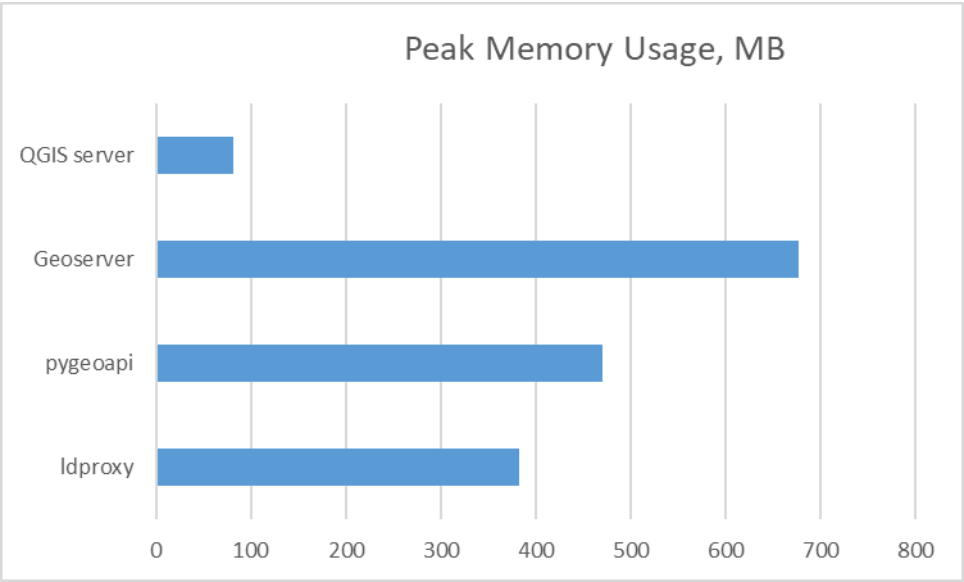


Figure 6. Peak Memory usage of QGIS server, Geoserver, pygeoapi and ldproxy.

In memory usage test, again QGIS server has best result with 82 MB, follow by ldproxy with 383 MB, pygeoapi with 470 MB and Geoserver with 677 MB (Figure 6). The advantages of native applications is obvious. Geoserver worst performance is due to its vast variety of functionalities (means lot of components are loaded) and especially its Java nature.

DISCUSSION AND CONCLUSION

Regarding achieved result several conclusions could be made:

- As an overall performance OGC API implementations shows better performance than traditional OGC Web Services ones.
- When it comes to best performance in terms of consumed resources native applications have no alternative. Considering installation and configuration processes Geoserver with its graphical administration interface is the best option as well as in terms of complete product and integration with other systems. For all other, third party solutions are necessary.
- All of the products functionality can be extended via plugins which makes them flexible enough for various kinds of use cases.
- There is no best option between four products. The choice is dependent on project goals, available resources, technical expertise of the stuff, etc.

As future work, this research can be extended in several aspects:

- Including additional products (including proprietary ones). Beside that QGIS and Geoserver have support for OGC API Features (in QGIS natively, while in Geoserver via plugin) so it is possible to be compared by this criterion.
- Use specialized tools (profilers) for precise measurement of consumed resources.

REFERENCES

[1]. Zhou, Chenghu. (2025). Exploring future GIS visions in the era of the scientific and technological revolution. Information Geography. 1. 100007. 10.1016/j.infgeo.2025.100007.

[2]. Chowdhury, S., Schröder, D., Ostadabbas, H., Hosseingholizadeh, M., and Friesecke, F.: Modernizing geospatial services: an investigation into modern OGC API implementation and comparative analysis with traditional standards in a web application, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLVIII-4/W12-2024, 11–18, https://doi.org/10.5194/isprs-archives-XLVIII-4-W12-2024-11-2024, 2024.

[3]. Sulaeman, L. F. P. and Behr, F.-J.: OGC API: Ldproxy, an Updated and Contemporary Approach to Geodata Provision in Disaster Mapping, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., X-5-2024, 197–203, https://doi.org/10.5194/isprs-annals-X-5-2024-197-2024, 2024.

[4]. Santana I., Davis Jr. C., Comparative performance evaluation of OGC API and OGC Web Feature Service, Proceedings XXIV GEOINFO, page 134 to 143, December 04 to 06, 2023

[5]. https://www.ogc.org/standards/

[6]. https://docs.docker.com/get-started/docker-overview/

[7]. Nurseitov, Nurzhan & Paulson, Michael & Reynolds, Randall & Izurieta, Clemente. (2009). Comparison of JSON and XML data interchange formats: A case study. 22nd International Conference on Computer Applications in Industry and Engineering 2009, CAINE 2009. 157-162.

[8]. Introduction to OGC Web Services, An OGC® White Paper, May 30, 2001

[9]. https://docs.qgis.org/3.44/en/docs/server_manual/faq.html

[10]. https://portal.ogc.org/public_ogc/compliance/product.php?pid=1872

[11]. https://docs.ldproxy.net/references/specifications.html

BIOGRAPHY

Nikolay Krastanov graduated from the Technical University of Sofia, Faculty of Telecommunications with a master's degree of Telecommunications engineering, Computer networks profile. He was a PhD student during the period from 2013 to 2017 in the field of cybersecurity and cryptography. Nowadays he works in Ellatzite-Med AD as a software developer. One of the long-term projects he works on is strongly related to GIS. Nikolay Krastanov is member of IEEE in Computer and Communications societies.