University of Tartu

Faculty of Science and Technology

Institute of Ecology and Earth Sciences

Department of Geography

Master's thesis in Geoinformatics for Urbanized Society (30 ECTS)

Comparison of visualization and data access methods in a dynamic web mapping context for large geospatial vector datasets

Bakhtiyar Garashov

Supervisor: Phd, Alexander Kmoch

Tartu 2022

Comparison of visualization and data access methods in a dynamic web mapping context for large geospatial vector datasets

Abstract

Web GIS as a concept evolved in the 90s and refers to the possibility of using internet technologies to store, process, distribute and visualize spatial data with the involvement of multiple components which are connected via the network. Advancements in the technologies have resulted in the amount of data that needs to be published on the web increasing dramatically which in the end can be considered Big Data. It brings the importance of choosing the most appropriate data access and visualization methods for the web-based GIS applications which is the objective of this study. To achieve this aim, 2 country-level large geospatial vector datasets for Estonia have been included in the tests. In the first part of the methodology, a certain list of data access methods have been analyzed while in the second part the main focus was to reveal the characteristics and efficiency of multiple data visualization methods and web mapping libraries. The generated results have revealed that OGC-defined data access methods have limitations in regards to the data size where the number of features should not exceed a certain upper limit while pre-generated vector tilesets-based data access and visualization approaches showed the most efficient outcomes when the input data is voluminous.

Keywords: Web mapping, Web map services, vector datasets, dynamic maps, performance analysis and metrics.
CERS code: P170 - Computer science, numerical analysis, systems, control; P510 - Physical geography, geomorphology, pedology, cartography, climatology

Suuremahuliste vektor-ruumiandmete visualiseerimise ja juurdepääsu meetodite võrdlus dünaamiliste veebikaartide kontekstis

Abstrakt

Mõiste “Veebi-GIS” tekkis 1990. aastatel ja viitab interneti kasutamisega seotud võimalustele, et salvestada, töödelda, jagada ja visualiseerida ruumilisi andmeid, mis hõlmavad mitmeid läbi võrgu ühendatud komponente. Tehnoloogilised arengud on toonud kaasa veebis avaldavate andmemahtude drastilise kasvu, millega on seotud suurandmete mõiste. Uurimistöö eesmärk on hinnata laialdases kasutuses olevaid andmete juurdepääsu tööriistade ja visualiseerimise meetodite sobivust suuremahuliste vektor-andmekogude jagamiseks läbi veebi. Uurimistöös on kahe üleriigilise vektor-andmestiku näitel analüüsitud andmetele juurdepääsu pakkuvate meetoditega kaardivaate loomiseks kuluvat ajalist kiirust ja võrreldud visualiseerimismeetodite ja veebikaardistamise teekide omadusi ja efektiivsust. Uurimistöö tulemused näitavad, et eelgenereeritud vektor-ruutvõredel põhinevad andmete juurdepääsu ja visualiseerimise meetodid on suurte andmemahtude korral sobivaimad, samas kui OGC-määratletud andmete juurdepääsu meetodite kasutamisel on liiga suure andmemahu korral vähesoovitatav.

Märksõnad: Veebikaardistamine, kaarditeenus, vektorandmestik, dünaamilisedkaardid, jõudlusanalüüs ja -näitajad.
CERS kood: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine; P510 - Füüsiline geograafia, geomorfoloogia, mullateadus, kartograafia, klimatoloogia

Table of Contents

Abbreviations

GIS	Geographical Information Systems
WWW	World Wide Web
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup language
OGC	Open Geospatial Consortium
GUI	Graphical User Interface
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
OSGeo	Open Source Geospatial Foundation
WMS	Web Map Service
SVG	Scalable Vector Graphics
WFS	Web Feature Service
WMTS	Web Map Tile Service
TMS	Tile Map Service
OSM	OpenStreetMap
REST	Representational State Transfer
API	Application Programming Interface
RDBMS	Relational Database Management Systems
XML	Extensible Markup Language

JSON	JavaScript Object Notation
GML	Geographical Markup Language
KML	Keyhole Markup Language
SDK	Software Development Kit
VM	Virtual Machine
GCP	Google Cloud Platform
SLD	Style Layer Descriptor

1. Introduction

Since the advancements in terms of technology and Geographical Information Systems (GIS) within the last three decades when the Internet - an important technological phenomenon has been introduced, web mapping is becoming more vivid, interactive and lifelike hence serving the true meaning of being dynamic maps. With the new releases of substantial updates to web mapping technologies occurring almost daily, web cartography and specifically dynamic mapping innovations are at an all-time high right now. As the amount of spatial data has increased dramatically and required to handle, process, and visualize them in a highly-efficient way, paper-based cartography is becoming a less-preferable solution and has been overtaken by the applications where interactivity with the end-user became a necessity.

Traditionally, web-based GIS systems have been implemented as monolithic and platform-dependent applications (Wong et al., 2002). However, with the development of the Internet and the WWW, GIS has evolved and adapted to this new environment (Shekhar et al., 2001). The Web GIS became a synonym for web information systems that provides the functionality of geographic information systems on the web through HyperText Transfer Protocol (HTTP) and HyperText Markup Language (HTML) (Shanzhen et al., 2001). Throughout the history of cartography, paper-based maps have been providing the basis for expressing spatial information and phenomena to a wider audience. However, like most of the other specific domains, the current velocity of achievements in technologies revealed itself in cartography serving the geographical data to users in a dynamic way that provides highly interactive, extendable, and customizable solutions.

Throughout the history of Web GIS, several standards, methodologies, and approaches have emerged in the field and it helps to come up with a wide range of solutions for a problem. However, having a plethora of specifications and techniques might be an obstacle such as interoperability. To make a worldwide standard and to prevent arising issues from variations, Open Geospatial Consortium (OGC) was founded in 1994 which consists of 510 companies, government agencies, and universities participating in a consensus process to define and improve publicly known communication and interface standards. As a result of its commitment to working with governments, the private sector, academia, and research organizations, OGC is formulating guidelines for the open standards (Reichardt & Robida, 2019).

Although it was not widely used until the launch of Google Maps, web mapping is perhaps as old as the Internet itself. An important development, it is usually dated to the early 90s, when the WWW was officially introduced and the first browser with a Graphical User Interface (GUI) was released. Google maps was introduced with dynamic mapping and more user interface tools that made it successful (Ramsey, 2013). Both public and private sectors are increasingly using web map applications. Since the advent of the internet, the use of geospatial information has become increasingly common in people's daily life.

Today the improvements in computer science and cartography showed a positive effect on the amount of collected spatial data from multiple sources which makes it possible to make data-driven decisions. However, such a large volume of datasets requires novel methods, ideas, and algorithms to improve the process of handling the dataset including storing, distributing, and visualizing. While the usage of web technologies increased every day, the methods of Web mapping should be improved as well to process large geospatial datasets whilst providing maximum usability for the users and preserving the performance. Using voluminous datasets in the web map environments should be achieved while providing the user the possibility to interact with the data as much as possible. Although raster datasets have been used in the field of Web GIS and cartography thoroughly, the modern web mapping approach requires it to provide data access and individual selection of features which is made possible by using vector data.

The problem of providing the best performance to the user has always been one of the main focuses of computer science overall. However, only very few studies in the field have been done with the reference to choosing the most appropriate data access and visualization methods in the geographical context. Most of the works on the topic are limited with only focusing on the evaluation of multiple Web mapping libraries and OGC-defined standards. Moreover, according to Giuliani et al., 2013 rather than providing an overview or guidance for measuring the performance of data services, most studies concentrate on the usability of OGC services, and performance metrics analysis still exists as a field of lacking knowledge. Considering all these aforementioned gaps, the study focuses on using large geospatial vector datasets as input and provides a reference with practical benchmark analysis and inclusion of modern data access and visualization techniques.

The study aims to provide a comprehensive evaluation of methodologies and approaches in the context of accessing and visualizing large geospatial vector datasets over the web by analyzing performance impact of the concepts of tilesets, rendering, and caching solutions. The following research questions have been addressed within this study:

1. What are the current state and the challenges in the web-based GIS systems from the perspective of accessing and visualizing big vector spatial datasets?

2. What are the most effective methods of data accessing for large vector datasets over the web?

3. What are the most effective methods of visualization for large vector datasets over the web?

Within the first experiment, a list of standardized data access methods has been examined which was not completely covered in previous studies and the results introduced so that based on actual performance analysis metrics it will be understood the effectiveness of data access methods for large vector datasets.

GIS combines geographic science, mapping science, and computer science and improvements in computing models are leading to constant changes in the design and application of GIS (Zhao et al., 2015). Throughout its history and track of development, GIS has benefited from computer science and it is worth mentioning that some of the common concepts are highly applicable in this domain. Considering this aspect, modern caching solutions such as tilesets-based cache and web proxy cache have been examined and the results were revealed to conclude the research of data access methods.

The visualization of spatial data has become an essential part of data-driven GIS development where the spatial vector dataset is voluminous, and it is the most efficient and direct way to comprehend data. The second experiment has been dedicated to the evaluation of multiple data visualization methods in the study. It is obvious that besides providing an effective method for accessing the spatial data, it is important to visualize the data in a way that is as smooth and

efficient as possible for the Web GIS clients to render for the user. Within this part, a comparative analysis has been dedicated to get detailed information about the client-side and server-side rendering and rendering capabilities of Openlayers and Mapbox GL JS. Each of these libraries has specific rendering capabilities and performance; by default, Openlayers uses a Canvas renderer whereas Mapbox GL JS uses a WebGL-based rendering approach which is known for high-performance visualization capabilities in both 2D and 3D rendering. WebGL is a rendering engine that is well-known for its performance when the data is large, and complex in terms of geometry while Canvas is a simpler and slower rendering interface and recommended to use only simple and less data rendering processes. Currently, only a few research studies which have addressed this topic have been published.

1.1. Problem definition

In the GIS terminology the spatial vector dataset is described as data containing elements that are represented by multiple geometry types such as points, lines, and polygons (Shekhar et al., 2014). Spatial vector datasets with high accuracy and extensive coverage, such as land cover, social networks, and more, have expanded widely, providing a strong potential to improve decision-making (Yao & Li, 2018). Although modern Web GIS methods are improved well to make it feasible to work with both raster and vector datasets, using vector datasets is more prevalent as it allows more customization and flexibility on the client-side such as rendering, styling, data access, and processing. In order to further enhance Web Mapping interaction, vector data should be fully integrated with Web Mapping (Gaffuri, 2012). In spite of the fact that both academia and industry have proposed and developed new concepts, algorithms, tools and applications to enhance the value of consumption of large vector datasets, significant challenges within the context of spatial data management and more specifically, accessing and visualizing the data still persist. Data management is also a more complex and broad domain covering data storage, index and query, data sharing, processing, and analysis (Siddiqa et al., 2016).

When the subject is a GIS system that has a number of potential users and manages a relatively large spatial dataset, performance is a critical and the most notable challenge for developers. Limited access to the high-speed network connection, incorrectly modeled application architecture, selection of unfeasible tools, and limited hardware conditions are the main aspects that cause bottlenecks for the applications when it comes to providing a smooth user experience.

In terms of both data volume and technical applications, big spatial data is probably one of the most active areas with the approximate growth velocity of 20% every year (Lee & Minseo, 2015). Furthermore, the number of works and studies focused on spatial vector data is limited due to the fact that it is inevitable to face challenges while sharing and acquiring the data (Guo, 2017).

The server-side technology and resources used (CPU, Ram, data storage) differ significantly. There are still some important issues when working with large datasets. Web GIS applications load speed depends on a variety of factors such as unoptimized input data, a high number of HTTP requests, an insufficient number of resources of the server (Netek et al., 2020). The main obstacle still is a performance issue as nowadays big vector web maps in web mapping environments are usually too slow and not usable (Gaffuri, 2012). For example, when the dataset is voluminous, simple WMS requests (for example, GetMap) could take a longer amount of time to render the result while for the WFS service the number of features should be limited for performance. It is important to deliver something meaningful to the user as quickly as possible - the longer they wait for the page to load, the more likely they are to leave before waiting for everything to finish. An application should be able to show the user at least the basic view of the page they want to see, and eventually, more content will be loaded. For users, an acceptable wait time for a web request is about 8 seconds (Corner, 2010).

Maps that are being distributed via the tilesets-based approach could significantly reduce rendering time. Using pre-generated tiles instead of processing data on the fly or in the browser may reduce client-side workload and improve performance. However, generating those tiles on the server can be costly in terms of time and resources if the amount of data is significant. Additionally, because most of the time the requested resource will be similar or the same with subsequent requests, server-side caching solutions are an excellent approach. In the past, only a few studies have addressed this topic and to the best of our knowledge none of them went into detailed comparative analysis across multiple tools.

As can be seen from the title of the study, taking into account the above-mentioned issues, the main objective is to focus on applications where large spatial vector datasets are input, and comparatively analyzing data access and visualization methods and provide a reference for future possible works on the same topic.

2. Theoretical background

2.1. The WWW and GIS

About three decades ago, one of the most important projects of the modern world - WWW emerged to enable communication between one or more clients or users. The worldwide network (or simply the Internet) has been an integral and necessary part of the communicative basis of modern social life since the 1990s. According to a survey by Internet World Stats, as of 2021, the number of global network users was more than 5 billion (Internet World Stats, 2022). The first ideas about global communication networks can be traced back to the 1960s, which began as a governmental military initiative (Leiner et al., 2009). The varied and complex social and technological transformations we witness today have their roots in the way the Internet has been developed through research grants from the U.S. Department of Defense's Advanced Research Projects Agency (Cohen-Almagor, 2011).

The main use of this technology was limited to military projects rather than global use as is the case nowadays. Unlike today, computers or machines had very limited capabilities and were used by research institutions, government agencies, and universities. To exchange information between them, a global network with a limited range was established. The network of computers was from the start an open, diffused, and multi-platform network that up until the 1990s developed in the United States and within a few years, expanded globally (Cohen-Almagor, 2011). Thanks to the advances in computer science and technology, the late 1980s became a milestone for the global network as personal computers and other devices that could connect to the Internet evolved. The main problem solved by the introduction of TCP/IP and other communication protocols was to provide a consistent way of communicating between different corners of the world over the network.

A **protocol** is a collection of preset rules that allows two or more participants of a network to communicate with each other and to facilitate Internet accessibility (McGarty, 2002) .The ideas of researcher Tim Berners-Lee at CERN (the European Organization for Nuclear Research) were the most important steps toward the modern Internet, which includes the specifications of the WWW, Uniform Resource Locator (URL), HTTP, and HTML (Jacksi & Abass, 2019).

The Internet grew in popularity and flourished as more agencies and individuals joined and pioneers pushed the boundaries of the platform with new, innovative ideas. Around the same time, modern clients appeared that could access endless resources through a GUI that became available on the global network. Large companies and organizations became aware of the benefits of the Internet and began to invest resources and time in sharing information about their services, placing advertisements, and publishing their websites on the Internet. To provide consistency, an organization called the World Wide Web Consortium (W3C) was formed to define global standards for the Web. Although the basic idea behind the web was to provide a solution for sharing and publishing documents on the web, nowadays it has much broader applicability and usefulness, such as social media.

In the 21st century, almost everything is digitized, even maps are no longer a special case. Every aspect of our lives has changed, from our everyday activities to our professional tasks. Among other things, GIS has been influenced by this phenomenon and a new technology called Web GIS or Web mapping has emerged. With its sophisticated analytics capabilities, web has revealed the tremendous value and awesome ability of GIS to quantify large integrated spatial data and it has introduced flexible architectures to make distributed geographic information accessible to a very large global audience using an advanced IT infrastructure. Web GIS provides a fresh perspective on information and brings to life the hidden meaning of data to reveal new opportunities and innovative approaches with an intuitive user interface (Tiwari & Jain, 2013).

Today, the majority of Internet users utilize this tool, but most do not even realize it. To put it simply, searching for hotels and addresses in a strange city using Google Maps or similar apps, and tracking your path using routing features are the best examples of use cases of this technology.

The basic idea of Web GIS is not so different from an ordinary web application. Data is exchanged between a GIS server, which provides its functions and resources, and a client, which can be a web, mobile, or desktop application.

Because of the WWW and customizable web services, an immeasurable amount of geospatial data is now available in various formats and can be used without any limitation. Apart from all these advantages of Web GIS, cartographers should undoubtedly have some basic knowledge of the web, its architecture, and GIS. As a result, professional cartographers have needed to update

their skillsets in response to shifting client requests, while educators have needed to rethink their approach in regards to teaching (R. E. Roth et al., 2014).

2.2. Web GIS architectural patterns

GIS architecture and application models have evolved in lockstep with the advancement of computer science and technology and the progression of computing models, from desktop GIS in the 1960s to web GIS in the 1990s (Alesheikh, Helali, & Behroz, 2002). It is generally accepted that the development of GIS is heavily influenced by computer science while also lagging behind it (Yang et al., 2010).

The crucial concept on which almost all modern web applications are based is the core client-server computing model. The exchange of information takes place between a server and a client, where the server is a GIS server and the client basically is a web browser, mobile application or desktop application which sends a request and gets a response to serve the user (Pullar et al., 2011). This particular architecture is also called the networking-computing model (Figure 1), as all communication between two or more participants is facilitated by either a local or global network. As can be seen from the term used, the server is the one that creates, delivers, or provides resources over the network, while the client sits at the other end of the connection and consumes the resources. The minimum requirement for a web GIS application is a client through which the application can be accessed, a server which shares the sources, and the ability to establish a connection between these components.

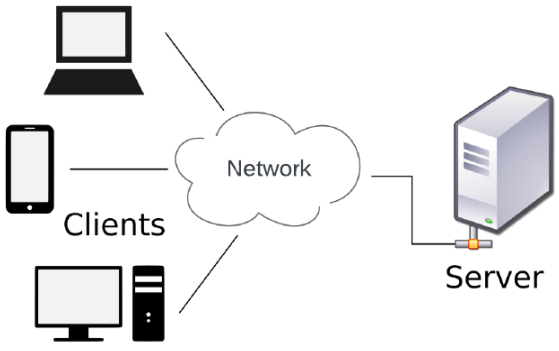


Figure 1. Core client-server architecture

The enabled communication between peers of this architecture benefits the TCP/IP stack. Upon establishing a successful TCP connection client and server communicates via the HTTP protocol which is used by clients to send requests to the server. The main method of data sharing or accessing over the web relies on the HTTP which gives the base for web service-based application implementations. Depending on the client's need there are 4 main types of requests (Table 1) where each of them performs a particular type of action on the server. Those requests are GET, POST, PUT and DELETE.

Table 1. Description of 4 main HTTP methods

HTTP method	Action	Description
GET	Read	Returns requested data
POST	Create	Creates a new record
PUT	Update	Updates an existing record
DELETE	Delete	Deletes an existing record

There are three types of architecture based on the weight of the components (Doyle, n.d.). Based on the responsible part in which the data is being processed we can divide Web GIS architecture: **thin architecture** where the server is responsible for performing each and every operation on the data and providing the client with end results. The opposite one is **thick architecture** where the client is responsible for processing all the data. The only input for the client is unprocessed spatial data. Main benefit of developing apps with thick architecture is that only cross-browser compatibility is required whilst for thin architecture applications are written in different programming languages and they are mostly heavily platform dependent where it is necessary to compile them every time when hosting on different platforms. Recompilation is crucial when there are differences between host hardware architecture and operating systems. The **medium client** approach is a hybrid wherein presentation elements (specified features and rasters) are known for being delivered to the client that is purely responsible for performing render operation.

Contemporary web mapping technologies, and web technologies generally, can be organized into one of three broad categories: (1) server-side technologies used to index and query geographic

information from a centralized source or, increasingly, distributed sources (e.g., the cloud), (2) client-side technologies used to render and manipulate web maps of this geographic information in the user's browser, and (3) web services or similar intermediary scripts used to relay information requests between the client and server (R. E. Roth et al., 2008).

Figure 2. End-to-end tiered Web GIS architecture

Numerous ways exist to publish map data via the web to make it globally accessible, ranging from static map files hosted on a very simple server to much more complicated architectural patterns where data is stored in the database, processed and served via sophisticated web or GIS server and all those components tightly coupled that are required to interact with each other. Almost all kind of modern applications heavily rely on tiered architecture (Figure 2) with 3 isolated yet strongly linked main layers of implementation: **Presentation tier** - a client application in which the users are interacting with directly, **Application tier** - a web server where data is obtained from the source and is being processed and **Persistence tier** - consisting of a data storage where data is being kept for long term.

2.3. Data access methods and standards

2.3.1. Open Geospatial Consortium - OGC

In the last few years, a plethora of web mapping specifications, methodologies, approaches and tools have emerged in this field. Over time, some of these initiatives have been deprecated already, some of them are not accepting support anymore, while some of them just don't have any recent contribution. A growing number of producer-operated map servers were established by the year 2000, but they operated in isolation with no integration of multiple services (Morris, 2006). Arising issues from this perspective requires having a body which is responsible for defining worldwide standards and specifications that are globally accepted. Standards for

geospatial data exchange developed by the OGC provide consistent foundations for the development of GIS systems (Michaelis & Ames, 2017). This organization defines three main standards that could be used in a Web based GIS application. These are the Web Map Service (WMS), the Web Map Tile Service (WMTS), and the Web Feature Service (WFS) (Li et al., 2011). Due to the diverse, huge, and complex nature of the geospatial data on which they operate, geospatial services differ slightly from common services. Among them are WMS for displaying maps as digital images and WFS for accessing geospatial features shared as GML and GeoJson (Lupp, 2008).

2.3.2. Open Source Geospatial Foundation - OSGeo

In February 2006, multiple organizations, individual contributors and maintainers of free and open source geospatial projects made a collective effort to create the OSGeo (Coetzee et al., 2020). OSGeo was created as a non-profit organization to stimulate the wider use of open source geospatial technologies, data, and training by collaborative development. Here it is important to emphasize the clear distinction between OGC and OSGeo; OGC creates and promotes the open standards of a wide variety of specifications whilst OSGeo is an organization to share and collaborate on open-source geospatial projects. Table 2 briefly describes the main characteristics of both organizations.

Table 2. OGC and OSGeo comparison

Organization	Idea	Important projects and specifications
OGC	Collaboration on defining open standards	WMS, WFS, WMTS
OSGeo	Collaboration on creating and promoting open source projects	GDAL, OpenLayers, GeoServer, MapServer, PostGIS

Each of these organizations has an interest in spatially related software, data, and standards, and recognizes the term "open" quite clearly from different perspectives.

2.3.3. Web map service (WMS)

The first WMS implementation specification was published by the OGC in the year 2000 (Brinkhoff, 2007). WMS is one of the most fundamental data delivery services for accessing and sharing spatial data in raster format which provides the specifications for georeferenced maps. The word “map” in this context expresses a visual representation of the data in the sense that it is not the data itself but the result which is rendered in pictorial format such as JPEG, PNG or occasionally, Scalable Vector Graphics (SVG) (Qian et al., 2004). WMS supports 3 types of requests which are described in Table 3 and Figure 3.

Table 3. Supported request types via WMS

Request type	Description
GetCapabilities	Returns service-level metadata to the client as XML document
GetMap	Returns the map in raster format. Usually the output is JPEG or PNG
GetFeatureInfo	Optional operation to fetch additional information that was included in GetMap response

The type of request should be included in the WMS request as an URL parameter. Besides the request type, there are more parameters available (Table 4).

Table 4. Mandatory parameters of WMS request (OGC e-learning platform, 2022)

Request parameter	Description
VERSION=	Request version performed by client
REQUEST=	Type of the request (One of the options from Table 3)
LAYERS=	Requested layer(s) from the server. Multiple values is also possible separated by comma
STYLES=	Comma-separated list of applied styles to the rendered result
CRS=	Coordinate reference system such as EPSG:3301
BBOX=	Bounding box corner values. MinX,MinY,MaxX,MaxY from lower left to upper right.
WIDTH=	Width of the map image (in pixels)

HEIGHT=	Height of the map image (in pixels)
FORMAT=	Output pictorial format of the map (PNG, JPEG)

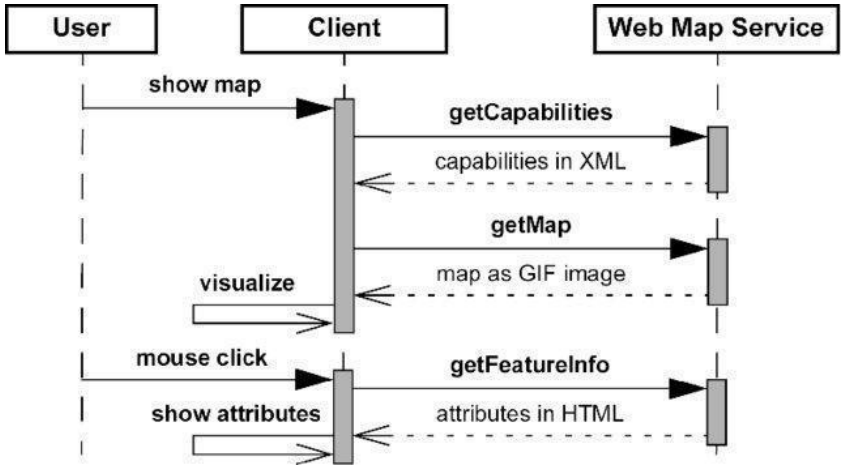


Figure 3. Life-cycle of a typical WMS (image credit: Thomas Brinkhoff)

Except the spatial data itself, many other additional parameters also could be included in response such as WMS metadata (e.g., version, layer name), size, format and more. In order to fetch the data from a WMS service a special request type called GetMap should be performed. Passing additional request properties (bbox) would be helpful to retrieve only demanded portions of the data (e.g., one or more layers as a comma separated list of values). It is also feasible to pass width, height, spatial reference system and even background color.

2.3.4. Web feature service (WFS)

WFS is another specification of OGC that provides a way for GIS servers to serve its spatial features over the network. Unlike the WMS, WFS delivers geometry and attributes to the consumers. Also, passing additional parameters to query the dataset on the server and return the result as response is another advantage of WFS for the developers. WFS works with a different text data format; by default it returns data as encoded XML. Unlike WMS, WFS provides the data in vector format which consists of geometric features such as points, lines and polygons (Michaelis & Ames, 2017). WFS supports GeoJson specification as output too which is a widely-used spatial vector data format. Unlike XML, GeoJson is a faster, resource-saving and

easier-to-parse alternative (Nurseitov et al., 2009). In this way, processing and rendering the response as vector data within the client application or browser is faster. On the client side, applications can perform spatial operations, queries, and filtering of the styles of each feature.

2.3.5. Web map tile service (WMTS)

The OGC WMTS is yet another standard that performs communication and exchange of stored spatial data between one or more GIS servers and client applications. The main characteristic of WMTS is the ability to acquire map tiles of a spatially referenced dataset using tile images with predefined content, extent, and resolution. Typically, it serves pre-generated tiles from the server, with the most frequently used ones stored in memory (cached) to reduce latency. WMTS offers flexibility to users that only required part of the final raster data could be fetched from the GIS server which is a crucial advantage in terms of performance and the amount of transmitted data. While WMS focuses on the development of custom maps and is ideal for dynamic data or custom-styled maps, WMTS sacrifices the adaptability of custom outline rendering for the versatility made conceivable by serving static layers (basemaps) where the bounding box and scales have been compelled to discrete tiles. The settled collection of tiles allows a WMTS to be implemented using a web server that effectively returns existing records. (OGC E-learning platform, 2022).

2.3.6. Tile Map Service and XYZ

Tile Map Service (TMS) specification is a particular data access protocol that defines the access method to the rendered cartographic data at a fixed scale. Unlike WMTS which is defined by OGC, TMS specification has been introduced by OSGeo. Both WMTS and TMS is a protocol used to fetch the tiles from the server based on a specific directory layout where numeric indices are used to specify a particular tile. The practice of tiling is considered an effective way to reduce the size of raster and vector map data, compared to rendering it as one result. Table 5 shows the main differences between WMTS, TMS and OpenStreetMap XYZ specifications. The main difference between TMS and XYZ is the flipped Y axis. Furthermore, it is worth noting that TMS has been a less-used specification for tileset-based maps and has largely been replaced by OGC WMTS.

Table 5. Comparison of WMTS, TMS and XYZ

Specification	Characteristics
WMTS	● Defined by OGC ● Possible to fetch metadata ● Y axis goes down from the top ● Indices start upper-left corner ● Indices cannot be negative ● Supports *GetFeatureInfo* request
TMS	● Defined by OSGeo ● Possible to fetch metadata ● Y axis goes up from down ● Negative indices possible ● No support for *GetFeatureInfo*
XYZ	● Popularized by OpenStreetMap ● Less specific standards ● Interoperability issues possible ● No metadata specification ● Indices start upper-left corner ● Y axis goes down from the top

Tile-based or **Slippy maps**[1] are among the crucial and widely used geovisualization techniques. XYZ tiles are a way of expressing mosaic tiles based on their offsets (x, y) and zoom levels (z). A large seamless map can be created by loading tiles as a user browses around an interactive map (Adnan et al., 2010). In most cases, without having a server the tileset can be stored in the machine's file system and served statically. Instead of obtaining the entire map at once, a tile-based client constructs the map by requesting particular tiles and then combining them into a map. There are 2 main attributes that exist for tilesets: zoom level and coordinate pairs of the tiles. Zoom level is the scale of the map and when zoom level is 0 the entire map fits on a tile which usually has size of 256x256 pixels (Adnan et al., 2010). For each zoom level increase, a single tile is divided to 4 tiles and a maximum zoom level of 22 is usually enough for the mapping purpose. Figure 4 depicts the tile division structure of tileset-based maps. At any zoom level, a specific tile is identified by a pair of numeric values which are cartesian coordinates. To

[1] A term introduced by OpenStreetMap to describe a tilesets-based interactive map where hundreds of individual tiles are merged seamlessly

sum up, each zoom level corresponds to a directory, each column is a subdirectory and each tile in the column is a file (OpenStreetMap contributors, 2015).

The directory layout is in the format of **/zoom/x/y.png.** An example raster OSM tile request looks like the following URL schema:

https://c.tile.openstreetmap.org/12/2352/1226.png

Figure 4. XYZ Tiling scheme

2.3.7. Modern REST service approach

Representational State Transfer (REST) is a suite of conceptual models that underlie the majority of the modern web and mobile applications with the classic client-server paradigm functioning as the fundamental architecture. Detailed information on how client and server communicate with each other via the worldwide network has been provided in earlier chapters. Aforementioned section also describes that the communication between those two parts is established based on defined standards of HTTP. When it is required to establish a standard technique of data interchange principles, the term REST comes in.

Given that REST is a phenomenon that specifies the acceptable and undesirable aspects of communication, it is more preferable to refer to REST as a collection of constraints rather than a conceptual or architectural model (Alimuddin et al., 2020). Building REST Application Programming Interfaces (API) is becoming ever more prevalent as microservices[2] and single-page applications[3] gain popularity. REST is an approach to supply consistent data exchange interoperability across one or more servers and clients that have already been built following specifications.

Most mainstream Relational Database Management Systems (RDBMS) currently offer storing spatial data and developing REST services that serialize relational data into text-based GeoJson representation, ensuring an uniform and reliable method to data transmission. Furthermore, OGC has introduced a specification for the RDBMS which is called **OGC Simple Features Specification for SQL**. This specification introduces an approach to store spatial data in the table of database where one row corresponds to precisely one feature and alongside other non-spatial properties, one or more geometry columns have been included to store the geographical data. Besides Point, Line, and Polygon, this specification includes many other geometry types such as LineString, MultiLine, MultiPolygon, MultiPoint and more (Open Geospatial Consortium, 2006b).

Under the data formats section, detailed information about JSON and its spatial extensions was given. Figure 5 provides an illustration of a GET request to fetch data in the GeoJson specification. It can be seen that the example returns a restaurant record with a number of properties. The main one which defines the appropriate GeoJson format and spatial vector data in general is the **geometry** which explains the type of geometry and gives a pair of coordinates in a list.

[2] An application which consists of multiple loosely-coupled services and communicate via network

[3] An application model where the content is being rewritten on the same page instead of loading the entire page for each changeset

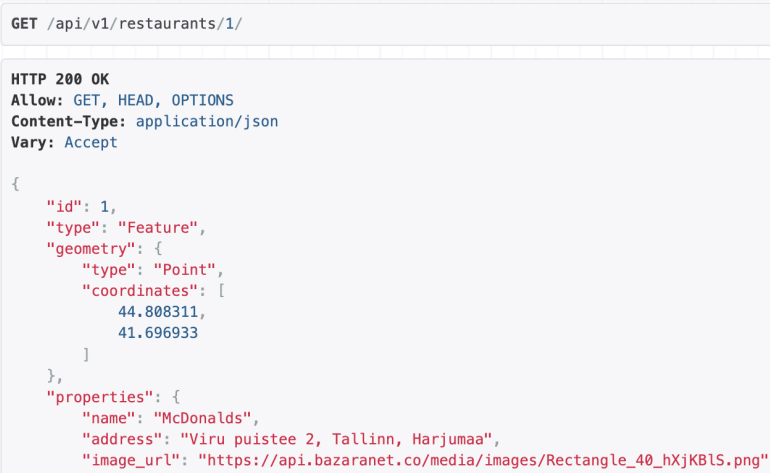


Figure 5. An example of HTTP GET request and response

2.4. Data formats

In the problem definition section, interoperability has already been depicted as one of the prevalent issues of current web-based GIS systems. A common way to achieve interoperability is to standardize the interfaces of functional components, data access and formats across different Web GIS systems. On the other hand, standardized data formats such as eXtensible Markup Language (XML) and JavaScript Object Notation (JSON) are the modern feasible standards to share data. For presenting and exchanging 2D location-aware information online, a variety of formats and standards are used (Behr et al., 2011). This part of the study focused on XML and JSON - more recently XML became the basis of many Web GIS OGC standards, while after the appearance of the modern REST services approach in the industry JSON usage increased drastically.

2.4.1. JSON and spatial extensions

Before moving to GeoJson or TopoJson, it is worth explaining the unified JSON format in detail. JSON format is a text-based means of representing structured data using JavaScript objects (Mozilla developer network, 2022). In web applications, it is typically used to transfer data from one server to another (when data needs to be sent from the server to a client to be rendered on a web page, or vice versa).

A variation of JSON, GeoJson, is commonly used in the geoinformatics community (Schaub et al., 2006). In particular, it can be used to combine geometric features or shapes. By using different geographic data structures, GeoJson simplifies the encoding of spatial data into JSON. Furthermore, a slightly modified version of GeoJson exists which is called TopoJson and is being used to encode topology characteristics of spatial data.

2.4.2. XML and spatial extensions

Modern GIS tools and OGC defined specification solutions commonly rely on XML to define data objects, data structures, assigning attributes and other data, storing configuration parameters and metadata; its functions are several. Like HTML and most other markup languages, the basic element of XML is a tag which defines a core structure. A tag also can accept one or more predefined attributes during the definition process like shown in Figure 6. XML supports nested structure which means that one element can accept another valid XML tag besides a value or a text.

In Geoinformatics, XML serves as a base for other derived text-based data formats such as Geography Markup Language (GML) and Keyhole Markup Language (KML). XML has been used by OGC standards such as WMS and WFS which have been introduced in the 2.3.3 and 2.3.4 sections respectively.

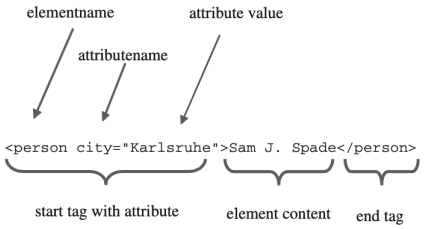


Figure 6. Structure of a XML element

GML is used to encode, describe and model geo-objects and features which are intended to be used as abstraction mechanisms of real-world phenomena in a data model such as a street, a road or a point of interest. Annex 1 contains an example sample object of GML format. It is an XML grammar that contains both spatial and non-spatial properties of geographical features, and is written in the XML schema for modeling, share, and storage of geographic information (Agrawal & Gupta, 2017).

Another extension of XML is **KML** which was firstly created by Keyhole, Inc. which had been obtained by Google in 2004 (Behr et al., 2011). KML is one of the de facto standards developed and used by Google Earth. The inclusion of this format in the Google Earth project was a move by Google to increase its popularity. The list of applications using KML today is extensive, including Internet mapping APIs, desktop GIS systems, and virtual globes. It is a limited format in the sense that it only supports the WGS84 coordinate reference system. Annex 2 illustrates the KML sample data.

2.4.3. Vector tiles

Tiles or tilesets are geographically pre-defined sets of squares which are easy to render on the client with less time and computing resources being used. For this reason, the performance of Web GIS applications might be increased noticeably. Currently, tiling the geographic data can be done in two ways:

- Raster tiles
- Vector tiles

For raster tiles, any changes in the dataset will result in re-generate tiles from source data while in vector tiles only geometry and attributes are stored in the server and all styling, symbology, and defining zoom levels are done in the client environment (Netek et al., 2020). Modern user experiences require dynamically changing data, styles, and symbology, and unlike raster tiles, which are generated images, vector tiles are simple vector objects that can be styled via client-side scripting. While using vector tiles is becoming widespread, today most of the basemap providers like OpenStreetMap use raster tileset-based solutions. Vector tile is a newer approach in the Web GIS and pioneered by Google for mobile and web versions of Google Maps in 2010 and 2013 respectively (Antoniou et al., 2009). The vector tile pattern and the directory layout to serve each individual tile are the same as it is in the raster tiles and has been given in the 2.3.6 section. The list below includes the key benefits of using vector tiles:

- Unlike raster tiles, styling can be applied dynamically by the map client
- Contains source data - spatial features that can be analyzed and processed
- Usually has lower tile size and faster response because contains less data

- Smooth and high-resolution results without bandwidth bottleneck because of client-side rendering
- Contains binary data which is faster to parse than text-based formats such as GeoJson/TopoJson

Mapbox is leading efforts to develop vector tiles and has introduced a specification[4]. Keeping data styling separate from its coordinates and attributes enables vector tiles to be more efficient. Using a set of predefined styling rules, clients can draw tiles based on vector coordinates and attribute data sent from servers. Vector tile's file format or extension is "pbf" while for raster it is "png".

2.5. Data visualization in the GIS context

It is obvious that expressing data in a meaningful way is as important as collecting, cleaning and processing the dataset. The increased use of GIS creates an apparently insatiable demand for new, high resolution visual information and spatial databases (Pundt, 2000). Data collection process often generates results in various formats such as qualitative, quantitative, hybrid and so on. In general, data visualization is a broad term referring to methodologies, processes, and the use of tools to create visualizations that can be communicated. The accuracy of obtained spatial data could be enhanced if the consumer can be able to instantly correlate the features to visual features on maps, satellite images, aerial photography, or other forms of 2D or 3D illustration. This refers to the basic paradigm of visualization, that "seeing" is a good way towards understanding, which finds support in the use of maps and other graphical displays (Hearnshaw et al., 1994).

Visualization is an excellent approach to improve the quality and efficiency with which spatial information is communicated to a wider audience. It should be as simple as possible so that non-specialists or specialists with backgrounds from diverse domains may understand it. Distributing geographic data over the WWW via Web GIS has a significant influence on data visualization as well. Unlike printed static maps, Web GIS allows for the creation of fully dynamic and interactive dashboards, maps, and other sophisticated visualization components.

[4] https://github.com/mapbox/vector-tile-spec

Web-based visualization approaches, as opposed to their predecessors, give considerably improved observability of the scenario. Besides all of its advantages, visualization results have to be evaluated extensively. It must be considered that one may be easily informed or misled by graphical display and that poorly designed visual displays may convey or even reinforce false ideas (Goodchild et al., 1996). In the analysis of mapping and communication models, maps can be considered as integral communication tools that help in decision making and spatial analysis by providing information about spatial phenomena. Before deciding on the description and graphics of the presentation, you need to determine the objective of the map, the target audience, the scale and the format to be used.

Turning to visualization of spatial information on the web environment, globalization has made the WWW an important tool for disseminating information to everyone, as the number of people connected to the internet around the world grows exponentially. The development of new technologies has led to a system that organizes products more quickly and allows for almost instantaneous interaction. They initially distributed documents with text and blurry photos in low-quality formats like GIF and JPEG.

2.5.1. Paper-based and digital cartography

Cartography is often defined as the art and science of making and using maps (R. Roth, 2013). Throughout the history of GIS, numerous approaches have been involved in the process of visualizing and introducing spatial data to the audience. While the initial versions of the maps were printed on paper after computer technology had been involved in multiple areas, cartography was not an exceptional case. A major reason for cartography that is being revolutionized is the relationship between geography and information technologies. Table 6 gives brief information about characteristics of using web maps and paper-based maps considering different characteristics.

Table 6. Comparison of digital and paper-based cartography

Characteristics	Digital	Paper-based
Storage	Easy to store	Requires to store physically
Data management	Can be automated easily	Needs to be reproduced to respond

		changes
Spatial analysis	Possible	Not possible
User interaction	Highly interactive and dynamic	Static and zero interactivity

2.5.2. Dynamic and static web maps

As it has been mentioned already, the term Web map is used to describe all types of maps that are available over the global network. Whether or not a web map provides the possibility for the user to interact with, they can be divided into 2 categories: static and dynamic. Static web maps are similar to paper maps in the sense that they are created once and are infrequently updated (Kraak, 2001). Editing or adjusting static maps mostly requires a considerable amount of time and effort and the finished result needs to be republished on the web to keep users up-to-date. Although most static maps are intended for web release, some are simply digitized (scanned) reproductions of hard-copy paper maps. Such maps are not always suitable for Internet use because the high resolution needed to accurately represent the quality of the original printed map often results in file sizes being too large to download (Peterson, 2003). It can be considered as a read-only method of publishing spatial data as neither there is a way of interacting with the map nor it is possible to personalize any part of it.

During the early stages of Web GIS applications, the result was mostly static maps. It means that sending a request to the dedicated spatial data server returns always the same response regardless of the attributes or parameters of the request. Strictly speaking, this should not be called GIS at all as it doesn't possess the ability to analyze geographical data (Soomro et al., 1999). A vast majority of websites include such maps that show locations and mostly they are using Google Maps for this purpose. Turning to advantages of static maps, reliability, simplicity, and performance are the main aspects. The major drawback of static ones is undoubtedly flexibility. Each time when the author wants to make any updates, he or she should go to the remote host server in which the application is stored and make appropriate changes to it. On the other hand, dynamic mapping on the web is a more flexible, modern, and advanced solution. However, it costs a lot in terms of both hardware, development and maintenance resources.

By contrast, dynamic maps are great tools to provide an interactive usability experience for the users with various kinds of map controls. Those controls can be basic zoom in or out, pan, layer selection switches which can easily be added via client-side scripting. Serving data via static maps is easy as it has been already generated (pre-generated) from geographic data in the form of a complete map, stored and ready to be served in the user's browser. However, for the dynamic experience, spatial data should be fetched from a remote source and the map generated on the fly in the client which should be done with some scripting in a programming language. Moreover, some advanced dynamic maps exist where it shows data or attribute changes interactively which are so-called animated maps. For example, time-based movement of storms or whales in the ocean could be some examples.

Static maps are the ones that provide Web map experience with minimum overhead in terms of computing resources. Those are simple files to be served in either the mobile application or browser of the user. As opposed to that, dynamic maps require Software Development Kit[5] (SDK) or libraries to be loaded on the client-side and being run by them. Creating a static map is as trivial as adding a couple of lines of scripting and some HTML tags such as canvas, iframe or image. Modern browser rendering engines or graphics APIs have been improved extensively so that rendering static maps will be very quick and provided quality of final results are seamless.

[5] A collection of software development tools as a single installable package

3. Study area, data and methodology

3.1. Study area

The study area of this research is Estonia, a northern European country that shares its borders with the Baltic Sea, the Gulf of Finland, and the Nordic countries.

In terms of geographical division, as of 2017, the first level administrative units of the country are counties ("*maakonnad*" in Estonian) (Figure 7). The main reason for choosing a country-level dataset is that one of the main requirements of this research is to analyze a relatively large amount of freely available dataset which will give a better understanding of main issues or limitations in regards to modern Web GIS data access and visualization methods.

Figure 7. First level administrative subdivisions ("*maakonnad")* of Estonia

3.2. Data

The experiments performed within this study include 2 vector datasets:

Estonian soil classification dataset

The digital soil map of Estonia has been produced with a total of 43300 km2 of Estonia's territory covered with data, except for urban areas, waters, and small areas without soil. The map has not been updated since the project ended in 2000 by the Estonian Land Board. However, significant efforts by the Department of Geography at the University of Tartu have been made to improve the dataset and most recent version published in November, 2020.

It is the most detailed soil map of Estonia with more than 750.000 soil units of the whole country mapped at a scale of 1:10 000 (Kmoch et al., 2021). In order to measure feasibility of the dataset in terms of volume, a couple of simple queries were used before final decision of choosing among other options. For example, a simple select all query for

*SELECT * FROM "EstSoil-EH_v1"."2" ORDER BY fid ASC*

fetching all the rows took 5 min and 31 seconds to complete and returned 745.432 rows which is really voluminous.

Data Output | Explain | Messages | Notifications

	fid [PK] integer	orig_fid bigint	upd_siffer character varying	wrb_code character varying	wrb_main character varying	boniteet bigint	varv bigint	loimis1 character varying	loimis_rec character varying	nlayers double precision	sol_zmx double precision
1	1	0	Ag	FL-gln	FL	0	15	[null]	I	1	1000
2	2	1	Ag	FL-gln	FL	0	15	[null]	I	1	1000
3	3	2	Ag	FL-gln	FL	45	15	[null]	I	1	1000
4	4	3	Ag	FL-gln	FL	45	15	[null]	I	1	1000
5	5	4	Ag	FL-gln	FL	0	15	[null]	I	1	1000
6	6	5	Ag	FL-gln	FL	45	15	[null]	I	1	1000
7	7	6	Ag	FL-gln	FL	45	15	[null]	I	1	1000
8	8	7	Ag	FL-gln	FL	45	15	[null]	I	1	1000
9	9	8	Ag	FL-gln	FL	0	15	[null]	I	1	1000
10	10	9	Ag	FL-gln	FL	45	15	$k^0_1sl/k^0_1ls_2$	$k^0_1sl/k^0_1ls_2$	2	1000
11	11	10	Ag	FL-gln	FL	0	15	krl/s	krl/s	2	1000
12	12	11	Ag	FL-gln	FL	45	15	krl/s	krl/s	2	1000

Figure 8. A sample dataset view from PostgreSQL interface

This kind of data is a perfect example to stress the performance and get extensive information about the limitations where modern Web mapping methodologies lack power. The dataset contains more than 50 attributes including soil type, soil classification, chemical composition and many more (Figure 8).

The dataset can be freely downloaded from Zenodo (https://zenodo.org/record/4291855).

Estonian Topographic Database

Although the Estonian Soilmap dataset is a perfect fit for the benchmarks, the unvarying nature of the dataset in terms of geometry types where all the features are polygon is considered as a limitation in the context of this study. For this reason, a dataset that contains a big number of features with multiple geometry types has been included which is the Estonian Topographic Database by Estonian Land Board. There is no limitation and the dataset can be downloaded free from the official website of the Land Board (https://geoportaal.maaamet.ee/index.php) in multiple formats such as geopackage or shapefile. Dataset contains multiple layers and the website includes a descriptive table[6] of them; the layer's original name in Estonian, the number of features and the description were given. It is worth noting that as the dataset includes 44 layers and the total number of the features are exceeding capabilities of processing for the device and most of the tools used in this study, only the layers below (Table 7) have been included.

Table 7. Selected layers from Estonian Topographic database

Layer name	Description	Feature count	Geom type
E_305_puittaimestik_p	Single tree, scattered trees, grove	560572	Point
E_305_puittaimestik_a	Forests	165140	Polygon
E_203_vooluveekogu_j	Watercourse (river, stream, ditch, channel)	879434	Line

Furthermore, it is worth to note that both of the included datasets are in the Estonian Coordinate System of 1997 (EPSG:3301)

[6] Layer names and descriptions https://geoportaal.maaamet.ee/eng/Spatial-Data/Topographic-Maps/Estonian-Basic-Map-1-10-000/ETAK-layer-names-and-descriptions-p710.html

3.3. Methodology

3.3.1. Pre-analysis

In order to make use of the geospatial data acquired, it must be processed and distributed first. Applications should be able to use the most up-to-date information by completing the processing in a reasonable amount of time. As the volume of data exceeds capabilities of regular machines in terms of processing, storing and sharing, geospatial data can be considered as Big Data (Kitchin & McArdle, 2016). **Cloud computing** technology addresses this issue with new distributed computing paradigms. Besides scaling, resilience, fault tolerance, and the ability to store and process large amounts of data, clouds are also highly responsive and can be accessed centrally.

Taking into account these advancements, the cloud environment has been extensively used within the methodology of this research. Since this analysis involves installing and deploying many geospatial services and working with them either jointly or separately, it is strongly believed that using isolated Virtual Machines (VM) to ensure exactly the same configuration across instances will allow benchmarking to run smoothly and reduce the chance of biased results caused by differences in computing hardware specifications. Since all these new approaches involve the use of cloud, the standard infrastructure of analysis which has been used in this study is Google Cloud Platform (GCP) which is one of the three main vendors on the cloud computing market today. Figure 9 describes pre-analysis preparation work which is divided into two parts: database and computing setup.

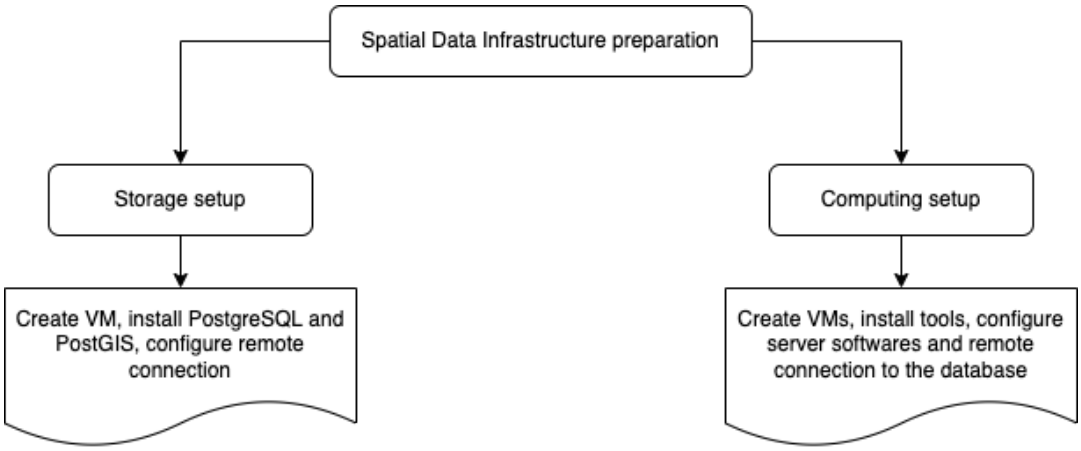


Figure 9. Pre-analysis preparation steps

Turning to the costs, since GCP offers free trial subscription credits in the amount of 300 dollars for the first 3 months, the whole analysis was free of charge. This is another main reason for choosing this product among other vendors.

The preliminary step of the analysis preparation was provisioning a PostgreSQL database with PostGIS spatial extension enabled. In spite of the fact that almost all cloud providers offer built-in computing resources designed to support RDBMS out of the box, creating a virtual machine and installing PostgreSQL/PostGIS manually seemed more straightforward and provided full control over database administration. With its 2 CPU cores, 4 GB of operational memory (RAM), and reasonable monthly estimated average costs, the ***e2-medium*** type of VM instance has been selected as it is one of the most appropriate variations of machine that GCP provides. Data Centers of Cloud providers are located across different continents, making them truly distributed and easily accessible from every corner of the globe. Reducing latency will be aided greatly by choosing a location that is close to the users as much as possible. Taking into account this, ***europe-north-1a*** has been chosen as the zone for all services which is located in Google's data center in Finland. After creating a Linux (Ubuntu) machine PostgreSQL and PostGIS can be easily installed by using the default package manager provided by the host operating system and configured for usage.

As it has been mentioned earlier, datasets are huge and it is not feasible to simply copy from a local computer into the newly created database in a cloud environment. That is why the GCP Object storage feature is used as a bridge between local computer and host machine. To put it simply, Object storage is a bucket that can be used to persist massive volumes of static data, folders or files and the public URL that it provides can be used to access the bucket content. Object storage has been also used to store vector tiles directory. Data preparation involves two main steps: creating PostGIS extensions for the connected database (1) and importing the geopackage dataset (2) using a tool called ***ogr2ogr***. The necessary commands for preprocessing steps can be found in the Github repository[7] of this study.

Finally, just as with the previous steps, provisioning the same type of machine instance is also necessary to host each software service and tool that will be used during analysis.

[7] https://github.com/Bakhtiyar-Garashov/thesis-everything

3.3.2. Experiments

The following sections will describe how test scenarios have been implemented, accomplished and what consequences have been made. In general, based on the nature of conducted tests the methodology phase of the study is divided into 2 parts: the first experiment will evaluate multiple OGC-compliant data access methods including WMS, WFS with GeoJson and GML, tileset and performance improvement of applying caching solutions. During the second experiment multiple data visualization methodologies and approaches will be examined. Table 8 describes the experiment scenarios in depth.

Table 8. Experiment scenarios

Scenario	Test cases	Expected results
Experiment 1	Data access methods: 1. WMS 2. WFS/GML 3. WFS/GeoJson Tilesets: 1. Vector tiles (Pre and on the fly generated) 2. WMTS Caching: 1. Tilesets cache 2. Proxy-based cache	● What are the best data access methods for large vector datasets ● How could tilesets improve the performance of accessing large vector datasets over the web ● How could caching techniques improve the performance of the applications
Experiment 2	1. Server-side and client-side rendering 2. Openlayers vs Mapbox GL JS comparison	● What are the most performant data visualization approaches for large vector datasets ? ● Server-side and client-side rendering: the performance analysis ● Openlayers and Mapbox GL JS - performance difference

Each one of the experiment scenarios requires a number of steps to process datasets, prepare the required infrastructure and installation, configuration of necessary tools and software packages (Table 9).

Results from each test case have been compiled into tables and for comparison a few graphs have been created to visualize the results in a more meaningful way.

Table 9. Test benchmark setup.

Machine	Operating system	Processor	RAM	Storage	Software
Postgresql/PostGIS database	Ubuntu 18.04 LTS	Intel Broadwell 6th generation, 2 vCpu	4 Gb	SSD disk, 50 Gb	PostgreSQL 11 with PostGIS 2.5 enabled
GeoServer	Ubuntu 18.04 LTS	Intel Broadwell 6th generation, 2 vCpu	4 Gb	SSD disk, 50 Gb	Open-jdk 8, GeoServer 2.19
Apache JMeter client	MacOS Monterey 12.3	Intel Broadwell 6th generation (i7), 6 Cpu	8 Gb	SSD disk, 256 Gb	Open-jdk 8, Apache JMeter 5.4.3

Finally, it is also crucial to note that the tests throughout the experiments of this study have been conducted with the high-quality connection and the important figures given below:

1. Upload speed: ~ 95 mbit/second
2. Download speed: ~ 90 mbit/second
3. Ping/Latency: 2-5 milliseconds

3.3.2.1. Data access methods

Within this section, predefined methods have been included to conduct test scenarios so that it will be clear what are the most effective methods to access large vector spatial datasets over the

web and how vector tiles and caching can affect the performance. Minimal initial styling has been applied during the tests in this experiment.

This part has been divided into 3 phases. In the first phase, OGC-compliant data access methods including WMS, WFS/GML, and WFS/GeoJson have been tested. In the second phase, 2 different scenarios (pre-generated and on the fly generated) with tiles have been performed so the tradeoffs between both use-cases will be clear. In the final phase, 2 well-known caching techniques are included in benchmarks which are tilesets-based and web proxy caching. Each of the included tests in this section will be performed utilizing 2 vector datasets which are introduced in 3.2.

OGC-defined data access methods

Multiple software tools exist to serve and acquire the geospatial data via the defined standards such as GeoServer and Mapserver. However, after the examination of both tools, it is worth noting that GeoServer has an obvious advantage in terms of usability, simplicity of installation, configuration, and user-friendliness with an easy-to-use graphical web-based admin tool while Mapserver lacks this feature. Moreover, taking into account that the idea of this study is performance analysis of multiple data access methods rather than tools only Geoserver has been included for the OGC-compliant data access standards.

For practical analysis, a couple of tests have been performed with the same datasets. Firstly, the usage of the datasets from different sources has been tested. Based on previous experience, it is known that thanks to advanced optimization algorithms (for instance, using indexing) being implemented in the background, using PostgreSQL/PostGIS powered database as a source performs better than reading a dataset directly from a vector data file stored in a machine disk (geopackage, in our case). In this regard, each tool has been tested via connecting to the PostGIS database and consuming vector datasets directly from the file. During the tests above, a specialized software - Apache JMeter for load tests is used as it is the most commonly used tool for a number of similar research projects. It gives a nice user interface (Annex 3) and it is possible to define the host address - where the requests will be sent, the number of requests, the possibility to save for sharing the test plan, and gives detailed metrics. All the performed test results have been saved as a spreadsheet table and included in the project repository. The schema of the tests is described in Figure 10.

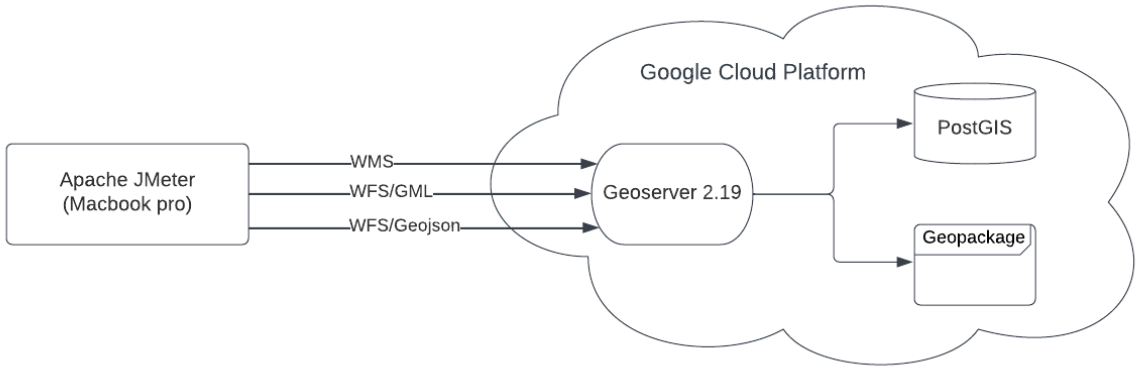


Figure 10. Load tests of data access methods via Apache JMeter

In terms of the number of requests in each case, it was necessary to perform statistically significant numbers to eliminate bias in the results as much as possible. JMeter makes it possible to use custom numbers so that 500, 2000, and 5000 sequential requests have been made.

Tilesets-based performance analysis

Within the context of this experiment generating and distributing data via the tilesets approach will be analyzed. There are numerous tools available today to generate both vector and raster tiles from a spatial dataset and even PostGIS natively supports the creation of tiles via "ST_AsMVT()" function. Tippecanoe, GeoServer WMTS also serve the same purpose. Table 10 gives an overview of the scenarios included within this phase. Also, OGC-defined WMTS has been used to pre-generate and distribute raster tiles (png).

Table 10. Tilesets-based test scenarios

Scenario	Software	Hardware/Tiling machine
Vector tiles pre-generated and served from directory	• Ogr2ogr to create geojson • Tippecanoe to generate vector tiles from geojson • Gsutil CLI to upload the directory to GCP static storage	2 vCpu, 4 GB memory VM with Linux Ubuntu 18.04
Vector tiles pre-generated and served via Tileserver GL	• Generation steps are the same above (output as mbtiles) • TileServer GL installed to serve	2 vCpu, 4 GB memory VM with Linux Ubuntu 18.04

	mbtiles	
Vector tiles on the fly generated and served via server	• PostGIS “ST_AsMVT()” to create tiles dynamically from the database • Python (Uvicorn) server to query and serve tiles	2 vCpu, 4 GB memory VM with Linux Ubuntu 18.04
Raster tiles pre-generated and served via server	• GeoServer WMTS to generate raster tiles ahead of the time on the server	2 vCpu, 4 GB memory VM with Linux Ubuntu 18.04

Figure 11 shows the core schema of tilesets-based test scenarios from the PostGIS spatial data source. The same test cases organization schema which has consumed the data from different sources was used in these scenarios; fetching tiles from directory statically as XYZ (1) and via served TileServer GL. Besides this, the average (mean) load time of a single tile for raster and vector has been noted down. It is worth mentioning that during all of the tests within this study no concurrency has been applied and the number of sequential requests made by JMeter was the same as it is in the previous test and only single tile performance analysis has been performed because of the reason that the generated result for all the tiles is not statistically reliable due to dynamically loaded nature of tilesets-based approach. It means the number of loaded tiles depends on the user’s interactions with the map such as zoom level changes or movements to different directions (pan) and using a load testing tool like Apache JMeter is not able to imitate user interactions.

Finally, it is worth mentioning that multiple scenarios have been implemented using both datasets as input, and the advantages and disadvantages of tile generation approaches will be obvious after this experiment.

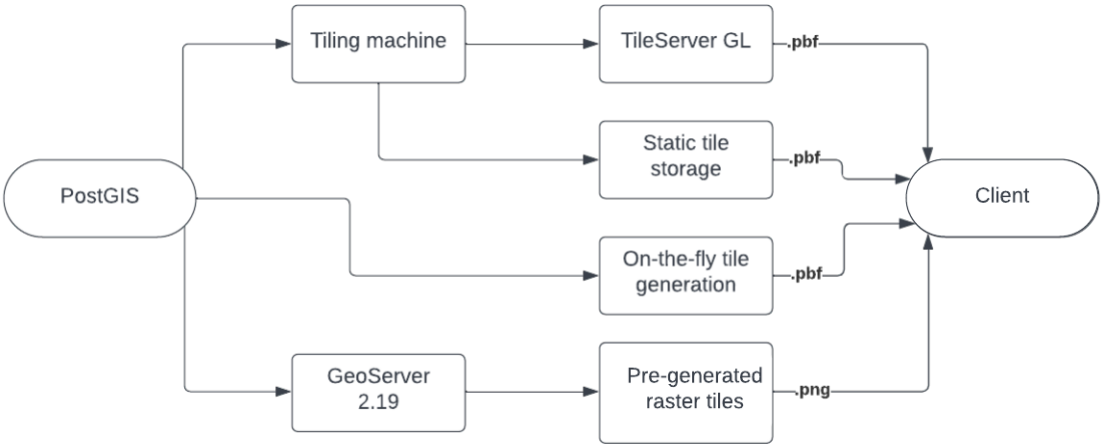


Figure 11. Tilesets-based test scenarios

Performance impact of caching

The demand for web maps increases year by year dramatically, it is important to maintain the performance of those applications consistently. Upon implementation of any application which is working with either spatial or non-spatial data caching plays an important role to accelerate the performance. Overall in computer science, many previous studies have depicted the caching methods, tools, and techniques for highly-efficient applications. Considering its success in the projects from different domains, this section of methodology has been used to demonstrate the benefits, methods, tools, outcomes, and potential problems of using caching techniques in GIS applications.

Two main methods of caching will be tested: web or reverse proxy caching (1) and tilesets-based caching (2). The necessary test metric within this part is response time and it will be compared with the non-cached WMS benchmark. Each of these techniques has its advantages and disadvantages and this section will try to emphasize them with practical results in the end. The methods of this study include using caching methods to increase the performance (reducing response time) of WMS GetMap requests which have been analyzed within experiment 1 and only the PostGIS database used as the data source, default WMS request schema and parameters such as bound box, dimensions of the result, output format (png 8 bit) have been preserved. 500 and 5000 sequential requests have been made with JMeter to assess the response time.

Dynamically generated content has become increasingly popular due to the demand for real-time information. Generation of the maps from the dataset on the fly is usually a costly process in terms of mainly time and performance and it made the developers consider caching methods to be involved. Upon establishing the successful HTTP connection, client and server share data within headers and body sections of request and response respectively. The header section plays an important role and defines the properties for concluding the information retrieval and exchange and caching header attributes for web caching are the crucial ones. The main HTTP-defined method for data caching is GET and browsers require cache headers to exist for the GET requests only. As it has been mentioned earlier, the WMS GetMap type of request is a simple GET request with a number of parameters such as style, bbox, width, height, and layer name included.

In terms of raster tile caching, it can be done using another OGC-defined service which is WMTS - a method of creating the tilesets ahead of time and distributing them in a highly-efficient way. The process of creating the tilesets beforehand is called **seeding** and it is the main step towards performance improvement. Additionally, it is also necessary to mention that although *Cloudflare* is another widely used general purpose cache tool for static files such as raster tiles for this study, the requirement of having an actual domain name for the server has made it to be difficult and considered as out of the scope of this test.

During the information exchange process between the client and server, the cache could be in one of the defined states which are included as a header in the response: HIT, MISS, or WMS. In simple words, if the data requested by the server is not present in the cached content and the request has been directed to the server to fetch it is called cache miss (Figure 12) while cache hit (Figure 13) is the action of fetching data from the cache instead of the server.

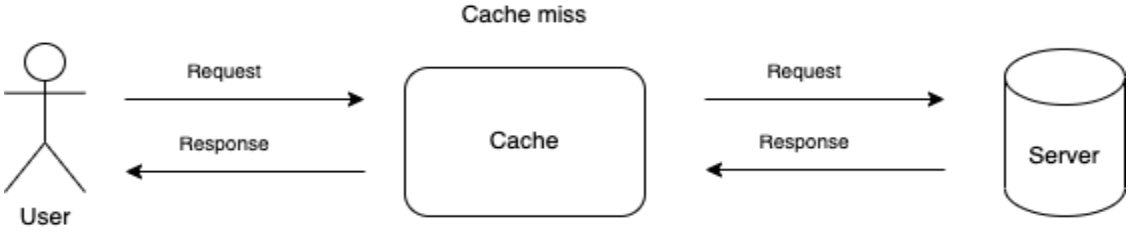


Figure 12. The request-response cycle during cache miss

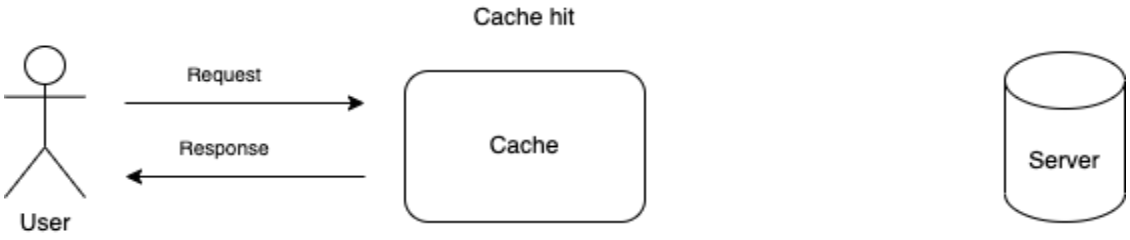


Figure 13. The request-response cycle during cache hit

In order to be aware of the caching process and states, it is necessary to have a look at main header parameters of caching. Table 11 highlights the main parameters that are necessary to examine for this experiment.

Table 11. Main generic cache header parameters

Header parameter	Description
Expires	A timestamp value after which the cache should be considered as legacy and should be re-generated
Cache control	Teaches the browser validate the cache based on specific provided value

Several tools exist for Tilesets-based caching and the built-in GeoWebCache service of GeoServer is an excellent approach. GeoWebCache is a tile cache within GeoServer and supports several formats including WMS-C which has been defined by OGC for the tile caching method for WMS requests. GeoWebCache commonly uses tiles with the dimensions of 256x256 and png-8 image format for the generated static images. An example URL for a successfully cached GeoServer WMTS request looks like Figure 14.

```
http://34.88.147.186:8080/geoserver/gwc/demo/MSc-thesis:2c?gridSet=EPSG:4326&format=image/png8
```

Figure 14. URL for GeoServer's GeoWebCache service.

Having a look at the response header (Figure 15) of this request we can make sure that the request was successful and the returned cache parameter is **HIT** which states that the end result has been acquired from the cache.

```
HTTP/1.1 200 OK
X-Frame-Options: SAMEORIGIN
geowebcache-tile-index: [291, 211, 8]
Content-Type: image/png
geowebcache-cache-result: HIT
geowebcache-tile-index: [291, 211, 8]
geowebcache-tile-bounds: 24.609375,58.359375,25.3125,59.0625
geowebcache-gridset: EPSG:4326
geowebcache-crs: EPSG:4326
```

Figure 15. GeoWebCache response header parameters

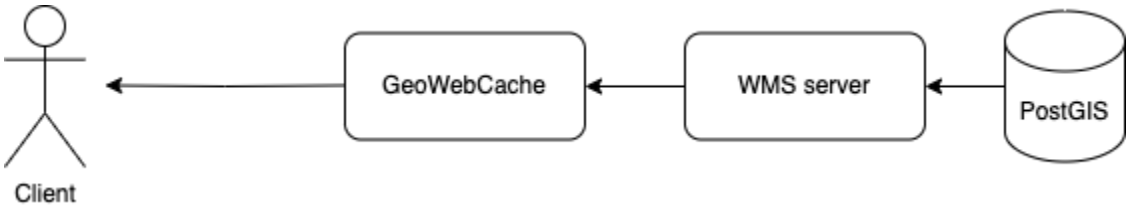


Figure 16. GeoWebCache's tile caching model

The next table gives information about each GeoWebCache's specific header parameters.

Table 12. GeoWebCache response header parameters

Parameter	Description
geowebcache-cache-result	One of the HIT, MISS or WMS values. HIT and MISS have been described previously. WMS means the data was not present in the cache and the request has been directed to the WMS source.
geowebcache-tile-index	The X,Y and Z values of the tileset. Z stands for Zoom level
geowebcache-tile-bounds	The bounds of tile in the corresponding CRS
geowebcache-crs	Simply the CRS definition of tilesets.

Considering the web or reverse proxy caching methods (Figure 17), some tools such as MapProxy, Varnish and Squid have been installed and configured for benchmarking reasons. All of those tools require initial manual work in terms of both installation and configuration. Before benchmarking, each of the tools has been evaluated based on their ease of use, installation, configuration, and the quality of documentation and the tabular result has been included within the results part. The value 1 to 3 describes the difficulty level (1 - easy, 2 - medium, 3 - difficult) of getting the specific tool up and running successfully. In terms of the learning materials and the quality of documentation, it will be evaluated with either A - means very straightforward or B - means lack of documentation. Following the flow of previous tests, 2 test cases have been examined during this part: WMS request consuming both datasets of the study. The average response time of each tool has been collected and compiled within the results section.

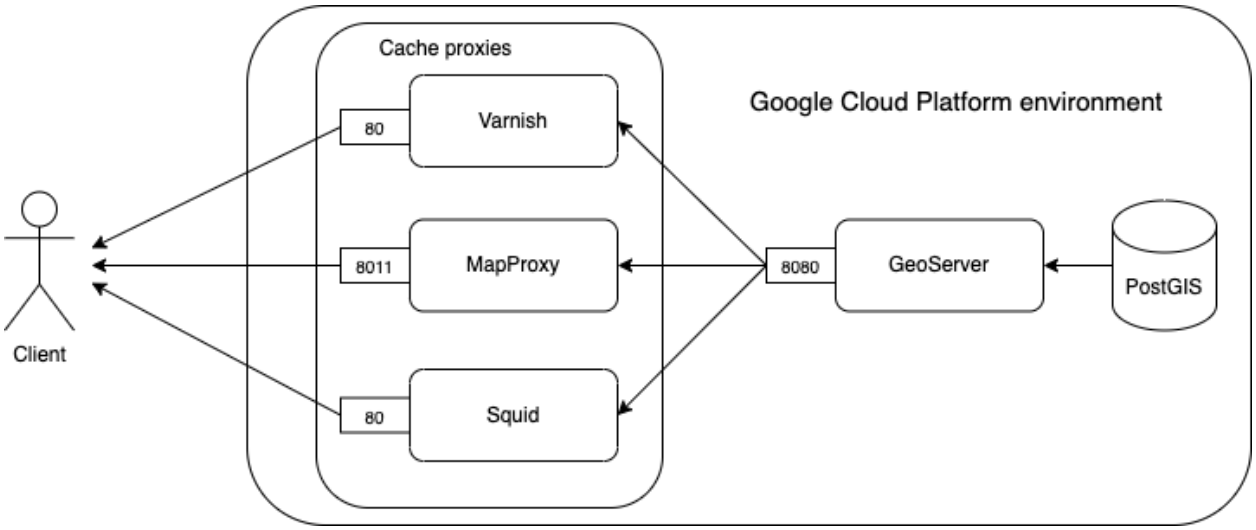


Figure 17. Web cache proxies core architecture

It is necessary to note that one of the previously created Virtual Machine instances, where GeoServer has been installed, was used for the testing during this phase. Although GeoWebCache comes installed within the GeoServer, other Web proxy cache tools such as Squid, Mapproxy, or Varnish require it to be installed and started in different ports of the machine separately.

3.3.2.2. Data visualization analysis

In the second experiment of the study, several tests have been performed to evaluate the visualization when the dataset is voluminous. The main focus of this part will be the evaluation of methods, client-side and server-side rendering, more specifically using WMS to generate, render and style the map on the server and using WFS/GML, WFS/GeoJson, and vector tiles where the rendering and styling have been done in the client environment.

For benchmarking purposes, Apache JMeter has been used with 500 and 2000 sequential requests. Before conducting the whole set of experiments, it is necessary to give some basic styling to the dataset to make it more understandable based on one or more attributes. After examination of attributes and their explanation based on the metadata file provided along with the dataset, the column for the definition of USDA texture classification ("lxtype1" on the actual dataset) has been chosen as it is one of the core representations of soil. GeoStyler plugin of Geoserver is a comfortable option for defining Style Layer Descriptor (SLD)-powered server-side WMS styling.

Before the benchmarks, it is necessary to make sure that included web mapping libraries within this part support the data access methods mentioned above. Today in the web mapping field, 3 main well-known libraries exist which are OpenLayers, Mapbox GL JS, and Leaflet.js. An initial capability analysis (Table 13) has been performed to see the exact availability of features for each of them as of 2022. Please, note that some of the libraries are actively maintained and in the future, these features might be changed. In the table each of the included libraries has been evaluated by either:

1. (+) Full support
2. (#) Support possibly by additional plugin, extensions or workaround
3. (*) No information or guidance found
4. (-) No support at all

Table 13. Web mapping libraries multiple data formats support

Data format	Openlayers 6	Mapbox GL JS	Leaflet.js
WMS	+	#	+
WFS/GML	+	*	#

WFS/GeoJson	+	+	#
Vector tiles	+	+	#

From the table, it can be seen that OpenLayers, which is a regularly maintained, mature, and high-developed web mapping library has native support for all of the included data formats and access methods.
In the case of Mapbox GL JS, it wasn't possible to identify whether there is support for WFS requests in case of output as GML either natively or via the help of a plugin or extension. Also, Mapbox GL JS supports WMS requests under limited conditions - only tiled WMS requests and EPSG:3857 coordinate reference system are possible.

Turning to Leaflet.js the built-in support for multiple data formats and access methods is extremely limited and can be extended only by installing additional plugins which are developed by open-source software developers. Considering these limitations and capabilities, for the performed benchmarks in this experiment Openlayers and Mapbox GL JS have been included.

As a side note it is worth to mention that in all cases only pre-generated Vector tiles have been included for client-side rendering in these experiments taking into account that in the first experiment it showed the best performance compared to on the fly generation of tiles.

4. Results

4.1. Data access methods

OGC-defined data access methods

This section describes the actual benchmarking results of the numerous tests which have been conducted during the practical part of the study. As the response time is considered the most important metric during benchmarking of a web-based GIS system, the results are worth focusing mainly on the timings. Additionally, the amount of received data has been included in the results too for WFS/GML and WFS/GeoJson.

Table 14 contains exact requests with all the included URL parameters. For the second dataset, it is simply enough to change the layer name to the included layers from the Estonian Topographic dataset.

Table 14. WMS, WFS/GML and WFS/GeoJson request

Service	Request
WMS	http://34.88.85.74:8080/geoserver/Msc-thesis/wms?service=WMS&version=1.1.0&request=GetMap&layers=Msc-thesis:2c&bbox=369013.875,6377148.5,736911.5625,6617856.5&width=768&height=502&srs=EPSG:3301&styles=&format=image/png8
WFS/GML	http://34.88.85.74:8080/geoserver/Msc-thesis/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=Msc-thesis:2c&outputFormat=text/xml; subtype=gml/2.1.2
WFS/GeoJson	http://34.88.85.74:8080/geoserver/Msc-thesis/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=Msc-thesis:2c&outputFormat=application/json

After making initial requests manually before introducing Apache JMeter it is revealed that to make a fair test in the case of WFS service for both datasets it was important to add *maxFeatures=500000* (where the number of features in a layer is more than 500000) so that the number of objects will be equal. For WMS requests, the default parameters such as bounding box, output formats were kept and all the features of the datasets have been included. Turning to

the main load tests performed via Apache JMeter, Table 15-18 and Figure 18 give an overview of the mean average response times for 500, 2000 and 5000 sequential requests.

Table 15. Estonian Soilmap dataset reading from PostGIS

	Number of performed requests		
Access methods	500 req	2000 req	5000 req
WMS	15.77 sec	14.63 sec	16.9 sec
WFS/GML	89.74 sec	119.1 sec	110.15 sec
WFS/GeoJson	79.91 sec	81 sec	83.54 sec

Table 16. Estonian Topographic dataset reading from PostGIS

	Number of performed requests		
Access methods	500 req	2000 req	5000 req
WMS	19.43 sec	21.76 sec	20.3 sec
WFS/GML	95.56 sec	103.8 sec	99.7 sec
WFS/GeoJson	83.4 sec	88.53 sec	86.87 sec

Table 17. Estonian Soilmap dataset reading from file

	Number of performed requests		
Access methods	500 req	2000 req	5000 req
WMS	17.5 sec	19.23 sec	18.4 sec
WFS/GML	108.76 sec	112.7 sec	105.56 sec
WFS/GeoJson	84 sec	88.98 sec	80.8 sec

Table 18. Estonian Topographic dataset reading from file

	Number of performed requests		
Access methods	500 req	2000 req	5000 req
WMS	24 sec	27.45 sec	25.09 sec

WFS/GML	113.45 sec	143.5 sec	132.47 sec
WFS/GeoJson	109.44 sec	103 sec	116.89 sec

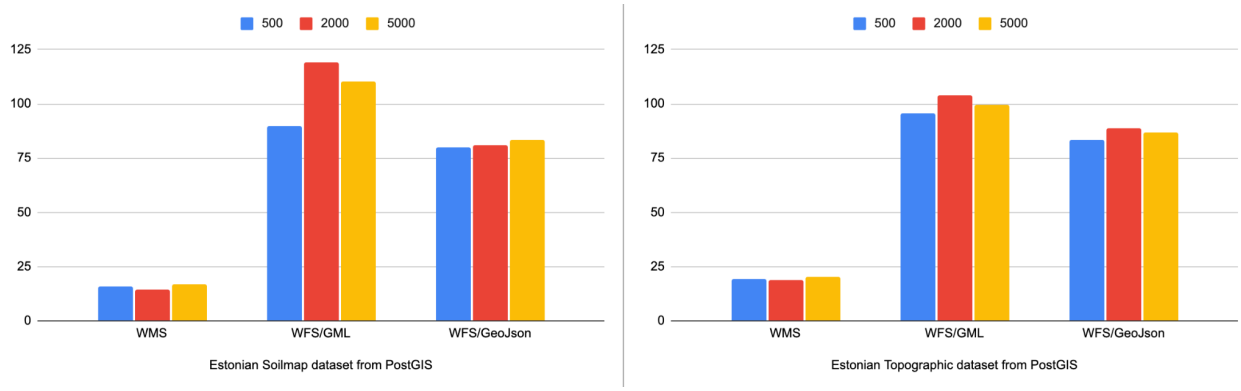


Figure 18. 500, 2000 and 5000 requests average response time in seconds for both datasets

It was expected that in the case of all three data access methods reading the data from the specialized spatial database has shown better performance. According to the number and charts given above, the WFS standard performs better in the case of the GeoJson as output format compared to GML. This might be the direct effect of the fact that GeoJson is a better data format compared to XML because of its resource-saving nature. To prove this theory, one more test has been performed to measure the amount of data received for both WFS GeoJson and GML. But in this case, the number of request features has been limited to the small numbers (100 and 10000) to see the variations. As the file size depends on the number of features, testing with multiple scenarios such as 500, 2000, and 5000 didn't have any impact on the results and a test with 1000 sequential requests has been conducted.

Table 19. Mean average received data size for Estonian Soilmap dataset

Request type	maxFeatures=100	maxFeatures=10000
WFS/GML	79.1 KB	14.8 MB
WFS/GeoJson	55.3 KB	11.5 MB

Table 20. Average received data size for Estonian Topographic dataset

Request	maxFeatures=100	maxFeatures=10000
WFS/GML	71.78 KB	17.8 MB
WFS/GeoJson	59.19 KB	10.9 MB

In the comparison of XML and GeoJson, it has been already mentioned that XML format tends to be more verbose and it results in sending more raw bytes than GeoJson in the cases above. And from Table 19 and 20 it is possible to see that the size of the data sent is much smaller when using GeoJson as output. For the exactly the same content, with fewer characters included GeoJson shows better results and it is important to note that the received data size is a crucial factor for the clients specifically with low bandwidth.

To sum up this test phase, it was revealed that for the large vector spatial datasets WMS offers better performance although unlike WFS, it has limited capabilities in the sense that the generated output is a raster and no interaction (select/access to individual feature) applies. In terms of vector outputs, WFS shows high performance if the number of requested features is approximately less than 5000. For the requests with more than 5000 features, the response time of the server decreases drastically. GeoJson showed better performance in all cases in terms of both average response time and the received data from the server in contrast to GML.

Tilesets-based performance analysis

In the 2nd phase, improvement of the performance via Tilesets-based data access and distribution approach has been analyzed. The main focus will be on the comparison of using **pre-generated** and **on the fly** or dynamically generated vector tiles and using OGC's WMTS to pre-generate raster tiles. The load tests have been performed using the same approach; Apache JMeter used as a tool and 3 different tests for each of both included datasets. Unlike WMS and WFS requests where the URL parameters (Table 21) were exact beforehand, for tilesets requests it was important to get the benefit of JMeter's random number generation feature to build the request URLs according to tile layout. Minimal styling via client-side scripting has been applied to the vector tiles to make the tiles visible. The initial test is a comparative analysis of 2 scenarios: pre-generated and on the fly generated tilesets. Table 22 briefly describes main characteristics of both.

Table 21. Different tiling scenarios requests

Scenario	Request pattern
Pre-generated vector tiles	https://storage.googleapis.com/eesti-soil-vtiles/soil_12c_drop_denser-8-16/{z}/{x}/{y}.pbf
On the fly generated vector tiles	http://34.88.85.74:8000/tiles/{z}/{x}/{y}.pbf
WMTS raster tiles	http://34.88.85.74:8080/geoserver/gwc/demo/Msc-thesis:e_305_puittaimestik_p?gridSet=EPSG:4326&format=image/png

Table 22. Pre-generated and on the fly generated tileset scenarios

Scenario	Preprocessing	Preprocessing time for 2gb data
Pre-generated vector tiles	● Generate Geojson from source data ● Generate mbtiles/directory tiles from Geojson	● Tile generation - 37 min ● Upload to storage - 12 min
On the fly generated vector tiles	● No pre-processing	● No pre-generation step
Pre-generated/pre-rendered raster tiles	● Configuration of WMTS ● Seeding the tiles (generation process)	● Installation of Geoserver and configuration - 15 min ● Tile seeding - 18 min

Having a look at the Table 22 we can see that for the pre-generated tilesets scenarios initial steps need to be addressed to generate the tiles ahead of time unlike on the fly generation process. However, as the main goal of this study is to mostly focus on the performance metrics such as average response time further benchmarks have been conducted. Table 23 and 24 give a detailed overview of the average load time of a single tile in multiple mentioned scenarios. 500, 2000, and 5000 sequential requests have been performed with both datasets.

Table 23. Estonian Soilmap dataset - single tile average loading time metrics

Scenario	500	2000	5000
Vector tiles pre-generated and served from directory	519 ms	503 ms	478 ms
Vector tiles pre-generated and served via Tileserver GL	232 ms	219 ms	225 ms
Vector tiles on the fly generated and served via server	1.3 min	1.47 min	1.26 min
Raster tiles pre-generated and served via server	172 ms	211 ms	232 ms

Table 24. Estonian Topographic dataset - single tile average loading time metrics

Scenario	500	2000	5000
Vector tiles pre-generated and served from directory	611 ms	583 ms	608 ms
Vector tiles pre-generated and served via Tileserver GL	392 ms	353 ms	398 ms
Vector tiles on the fly generated and served via server	1.56 min	1.62 min	1.546 min
Raster tiles pre-generated and served via server	168 ms	185 ms	172 ms

From both tables above it can be seen that serving the vector tiles via a special server overcomes clearly the case where tiles have been stored and statically served based on directory layout in performance. The main reason would be that while Tileserver GL is fast enough to traverse or walk through each directory, compute the particular tile to serve, for static storage there is no

ahead of time computation can be performed. Furthermore, multiple mixed geometry types in the second dataset didn't affect the overall performance of each scenario noticeable.

Turning to the on the fly vector tile generation scenario, this approach was extremely slow. For every request, the server receives z, x, and y values and makes necessary requests to the database to fetch data, build the tile and return. It is worth noting here that the application server and programming language nature might have a performance impact here in terms of response time too. However, evaluation of this aspect is considered as out of the scope in the context of this study.

Finally, the raster tiles always showed better response time on average. This is a direct impact of the fact that while the raster tiles are 256x256 images rendered on the server with tiny size variations because of the color heterogeneity while vector tiles take more time to load individual tile because of having actual attributes and geometry which makes it possible to get access to the individual feature on the map.

Performance impact of caching

In terms of experimenting and benchmarking different widespread caching tools and methodologies, it is important first to get the infrastructure ready and all the required tools installed and configured. As the methodology describes that it is planned to evaluate considered tools in terms of ease of installation, configuration, and having a high quality of guidance or documentation, the table below will give a brief assessment of each of them.

Table 25. Assessment of cache tools based on specific characteristics

Cache method	Installation	Configuration	Quality of documentation
GeoWebCache	No installation, pre-installed with GeoServer	1	A - detailed and straightforward
MapProxy	2	1	A - detailed and straightforward
Squid	1	3	B - lack of configuration guide
Varnish	2	1	A - detailed and straightforward

It can be seen from Table 25 that in terms of installation, the process has been comfortable for all of the involved tools. And as the Geoserver has been already installed in the VM instance, GeoWebCache comes builtin. Installation processes for MapProxy and Varnish have been graded with 2 (medium) as both tools have prerequisites such as Python programming language and Apache web server installed respectively. Turning to Squid, using the operating system's packet manager provides a straightforward way of installation. In terms of getting the tools in the working state, like most of the previous used ones they require some kind of manual configuration. While GeoWebCache, Mapproxy and Varnish provides an extremely detailed guide about post-installation configuration, the process for Squid was found highly tricky and tough in general and only some open source resources, and unofficial blogs helped during the tests.

For the benchmark results (Table 26) and performance analysis of those tools, each of the tools has been installed and used as a cache mechanism for the existing WMS setup from the first experiment. For the tile cache, GeoWebCache has been activated and configured while for the web/proxy caching the tools have been installed, configured, and tested separately. Before collecting the results it is important to make sure that the caching actually works and the spatial data have been fetched from the cache. Here the browser's developer tool feature is extremely helpful as it can be seen clearly whether the response for a particular request was a cache **HIT** or **MISS**.

Table 26. Non-cached vs cached WMS mean response times (seconds)

Estonian Soilmap dataset					
Requests count	Non-cached WMS	GeoWebCache	MapProxy	Squid	Varnish
500	16.9	2.3	4.8	5.87	6.7
5000	16	2.1	5.56	5.04	4.54
Estonian Topographic dataset					
500	18.6	4.3	7.6	10.7	9.54
5000	17.56	2.98	5.6	8.86	7.3

From the benchmark results table (also Figure 19), we can see that application of both web-proxy and tilesets-based caching increases the performance of WMS GetMap requests significantly for both of the large vector datasets. More precisely, in both cases, GeoWebCache gave the best result with 2-4 times less time than the rest of the tools. This might be the direct impact of the fact that GeoWebCache ahead of time generates the raster tiles with 256x256 dimensions and adds the most frequently request ones to the internal cache layer so fetching a tile from the cache is much faster than getting the whole server-generated raster result which is the case in web-proxy cache tools. Moreover, for the proxy-based cache mechanisms, MapProxy showed better average response times whereas Squid and Varnish had quite a high degree of similarity.

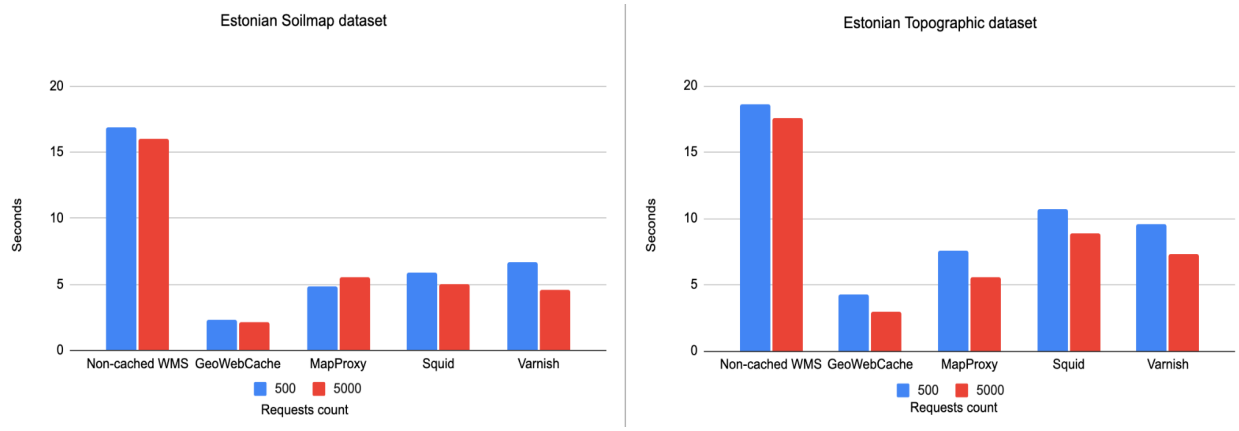


Figure 19. Non-cached and cached WMS request benchmarks (average response times) for both datasets

4.2. Data visualization analysis

As only Openlayers has support for the WMS GetMap type of request from the selected web mapping libraries, an initial test has been performed to evaluate the impact of styling. Figure 20 shows the styled WMS map of the Estonian Soilmap Dataset.

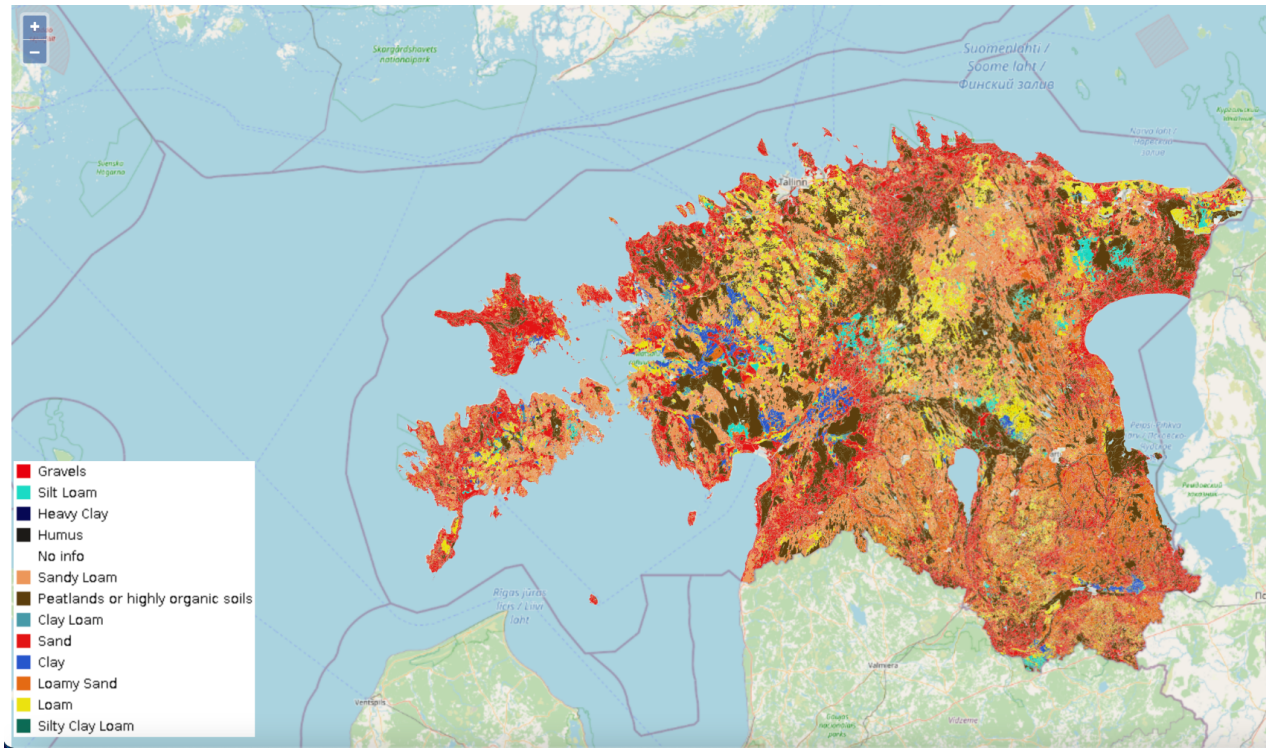


Figure 20. Estonia Soilmap dataset classified based on soil texture

Table 27. Non-styled vs styled WMS average benchmark results in seconds

Estonian Soilmap dataset		
Scenario	500	2000
Non-styled	15.68	16.32
Styled	16.54	15.11

It can be seen from the generated test metrics (Table 27), generated results are not aligned in the statistical sense and styling doesn't seem to have a noticeable impact on server-side rendered data visualization.

In terms of client-side rendering of the data, Openlayers have been used for WFS/GML, WFS/GeoJson, and vector tiles while without a guide or documentation for GML, Mapbox GL JS usage is limited to only WFS/GeoJson and vector tiles. Table 28 and 29 briefly shows the overall performance metrics of both OpenLayers and Mapbox GL JS libraries when consuming multiple data formats. It is also worth mentioning that for the HTTP timeout reason the maximum number of features for both WFS/GML and WFS/GeoJson has been limited to 5000 (maxFeatures=5000). Otherwise, the result was not shown in the browser due to the limitation of long-lasting requests in the modern browsers.

Table 28. Openlayers benchmark results

	Estonian Soilmap dataset (polygon)		Estonian Topographic Dataset (linestring)	
Data format	500 req	2000 req	500 req	2000 req
Vector tiles	516 ms	487 ms	552 ms	678 ms
WFS/GML	58.34 sec	55.3 sec	5.64 sec	7.56 sec
WFS/GeoJson	27.35 sec	24.56 sec	991 ms	898 ms

Table 29. Mapbox GL JS benchmark results

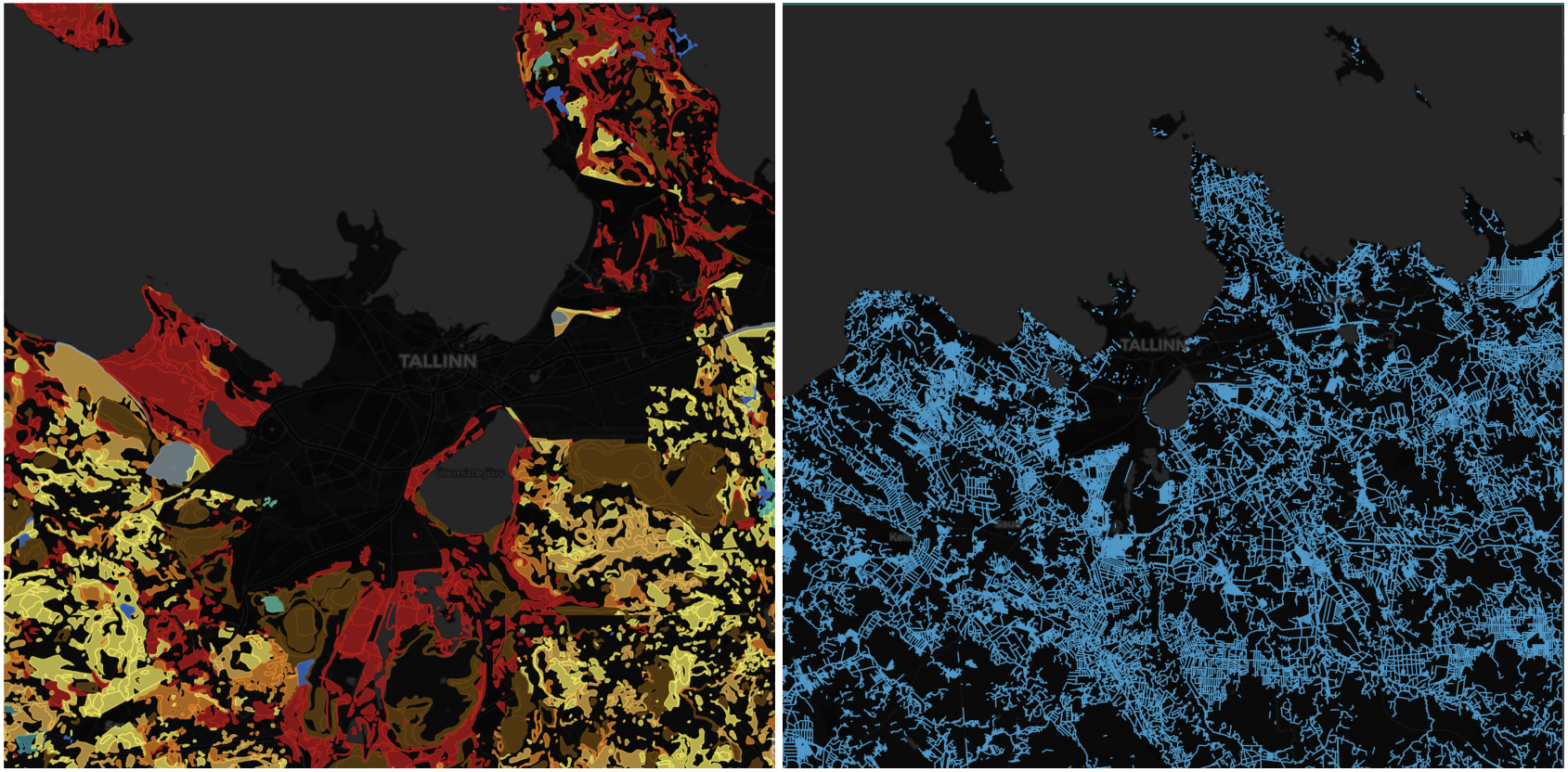
	Estonian Soilmap dataset (polygon)		Estonian Topographic Dataset (linestring)	
Data format	500 req	2000 req	500 req	2000 req
Vector tiles	318 ms	335 ms	298 ms	302 ms
WFS/GeoJson	23.1 sec	22 sec	765 ms	825 ms

First of all, it is obvious from the results that during the client-side rendering Mapbox GL JS showed noticeably better performance in all cases compared to Openlayers. This is a direct impact of the fact that Mapbox GL JS uses a WebGL renderer by default for every operation while OpenLayers uses Canvas 2D and WebGL-based rendering is only possible after manual configuration.

The next important revealed fact is that for the same amount of queried features (5000 features) geometry makes a huge performance impact. For the Estonian Soilmap dataset where all the geometry types are polygon, the average response time took more than 50 seconds while for linestring geometries the average result was less than 10 seconds which is the case in the Estonian Topographic dataset (layer is E_203_vooluveekogu_j).

To conclude the data visualization methods, it is an obvious fact that with its sophisticated rendering capabilities WebGL-based library would increase the performance to a large extent when the spatial vector dataset is huge. While the WMS has seemed the best method when it is required to generate, style, and render the result on the server, for client-side rendering the situation is complex. If the number of features does not exceed a certain threshold (~ 5000) WFS type of data access with GeoJson output gives better results compared to the GML format.

On the other hand, pre-generated vector tiles-based data distribution and visualization methods (Figure 21) showed the highest performance in terms of average response time which is measured in milliseconds. And main strength of vector tiles (or tiles in general) is the fact that while other vector data formats such as GeoJson and GML send the whole requested data to the client in one response and start the parsing and rendering process there, vector tiles take advantage of the tiles where each tile is asynchronous loaded small batches of spatial data and being downloaded dynamically only when requested by the user. And one more necessary fact is that the binary protobuf (.pbf) format is much faster to parse in the browser compared to text-based formats.



a) b)

Figure 21. Vector tiles - Estonian Soilmap (a), Estonian Topographic dataset (b) (layer: water course)

5. Discussion

The term "WebGIS" has come to stand for Web information systems that deliver geographic information system capability via the Internet using HTTP and HTML (Shanzhen et al., 2001). Today, this is a term used quite often in the realm of Geographic Information Systems. As the Internet rapidly grew in popularity in the 1990s, this phenomenon evolved. To share spatial information with the public in a sophisticated manner, spatial analysts, cartographers, and geographers began to explore the capabilities of this novel technology. The main reasons for its popularity are the new opportunities and workflows that Web GIS provides over the traditional application-constrained GIS system which is easily accessible, inexpensive, pervasive, and widely used. Web services allow us to connect to a world of information, from our enterprise to the Internet of Things (IoT), Big Data, and more. Due to a large amount of spatial data available via web-based architecture and the number of new tools required for their deployment, many questions arise regarding tools' efficiency, response time, suitability, and compatibility.

The amount of spatial data gathered has increased dramatically such as land cover, social media, mobility, and more which provides an opportunity of improving the quality of the decision-making process. At the same time, it is becoming a challenging process to manage the processing of data. Due to the ease of transmission and the well-established nature of the technologies, most Web maps are raster-based today (Antoniou et al., 2009). However, it presents several problems which arise from the limitations of raster datasets. A major advancement in cartography, the use of maps and vector data on the Web provides many new opportunities (Neumann, 2008). Thus, more and more studies should emphasize the usage of large geospatial vector datasets as input for modern Web-based GIS applications. Besides the advantages, the inclusion of vector datasets brings some challenges to modern cartography and especially web mapping. One reason for the lack of research on vector map data is that it is complex to simplify, process, and transfer over the web (Yang et al., 2007). The study of vector data transmission and visualization techniques is crucial since it may be inadequate to use raster images in some cases (Antoniou et al., 2009) or that vector graphics provide advantages for interactive maps over raster maps (Carpendale, 2003). Considering these advantages of vector data, the main objective of this study is to focus on the efficiency of multiple data access and visualization methods of the voluminous vector datasets over the web.

In the first experiment, OGC-defined WMS, WFS/GML, and WFS/GeoJson data access methods have been included to reveal the performance of each of these standards and the best methods for accessing large geospatial vector datasets over the web. From the results, it was obvious that while WMS showed better performance in terms of server-side generating and rendering the vector dataset in raster format and delivering to the user, WFS showed its limitations in case of the number of features exceeding a couple of thousands. If the dataset exceeds this limit, the waiting time for a simple request takes an extremely long time which is not acceptable in modern web applications. Furthermore, it is worth noting that modern browsers and HTTP protocol have an upper timeout limit, and exceeding this limit is another downside of long-taking data access methods which was the case in accessing data via WFS.

Turning to the benchmarks of WFS with multiple vector output formats such as GML and GeoJson some notable learning points have been revealed; for both of the recorded test metrics (average response time and size of received data), GeoJson as output showed better performance compared to the GML. This is directly connected with the nature of both text data formats. In the data formats section where dedicated information on both of them has been given, it is mentioned that XML tends to be more verbose than GeoJson. While the GeoJson format is a simple structure with the data described as key-value pairs and uses fewer bytes to describe the equivalent data, XML contains more raw data and consequently takes more space. Due to this size difference, it is revealed that the average response time is lower in WFS/GeoJson compared to WFS/XML.

Within the first experiment, a number of test scenarios have been conducted using tilesets-based data distribution. It was obvious from the benchmarks that using pre-generated tiles was extremely faster compared to the scenario where tiles have been generated on the fly from the PostGIS database. Dynamic (or on the fly) tile generation process is extremely costly as the server is responsible for accepting the z, x, y values from the client on each request, querying the database, and running PostGIS's ST_AsMvt() function to generate the vector tiles and send them to the user whereas pre-generated tiles showed much faster average response time while it has considerable ahead of time tile generation steps from the input vector data. Furthermore, it has been revealed that serving the pre-generated vector tiles via specialized software (TileServer GL, in our case) performs better compared to using static storage to serve the tiles based on directory layout. This is an effect of the computation which is being done by the software to compute the

exact tile for each request while for the static serving method there is no component to compute each specific tile and serve.

Lastly, two main methods of caching were explored: web or reverse proxy caching and tilesets-based caching. It has been already stated that whilst tilesets-based caching uses pre-generated tilesets so-called seeding from the original dataset and distributing them via OGC compliant WMTS request, web or reverse caching is a mechanism where the cache tool resides in front of the GIS spatial server and proxies the incoming requests. Upon accepting the same request a couple of times, the proxy tool is smart enough to cache the result of requests within its persistence layer and serve the data. For the initial requests map tile caches provide better performance as the so-called cold start for them is pretty quick whereas web proxy caches need to receive a couple of requests to persist that specific data to the cache layer. It is important to note that while map tile caches generate the tiles ahead of time and simply serve them, web caches save the data into their cache layer and fetch from there. The first result has overviewed the tools included in this test from different aspects: GeoWebCache was the one that does not require separate installation and comes bundled with GeoServer whilst other web proxy cache tools have the learning curve of the installation process. In terms of configuration and the quality of documentation, Squid was behind others and the official documentation lacks configuration and makes it possible to use. Overall, although each caching method showed different average response times, it can be concluded that applying any kind of caching mechanism has noticeably performance improvement in Web GIS systems and should be considered while developing applications where the performance is important.

To address the third research question where it was necessary to reveal the efficient methods of large vector data visualization, a number of test scenarios have been performed. Firstly, in terms of server-side rendering, simple WMS GetMap request tests have been re-examined to evaluate from a different perspective which is the possible impact of styling. However, during the tests no statistically meaningful results have been collected and seemingly there is no direct impact of styling in the case of server-side rendering.

Turning to the client-side rendering, a capability analysis was conducted to choose the web mapping libraries among many options in the industry. While Openlayers and Mapbox GL JS seemed highly supportive in terms of multiple data formats as input, Leaflet.js was excluded

because of the reason that most of the used data formats and access methods in the study are only possible via additional plugins and extensions. During the benchmarks, similarly to the results in data access methods, it was revealed that if the number of features is not exceeding a certain number WFS/GeoJson can be considered the better client-side data visualization method compared to WFS/GML.

It is important to note that while using Canvas as the default renderer, Openlayers showed relatively lower performance in terms of client-side data visualization in contrast to Mapbox GL JS which uses WebGL as rendering API by default. In the case of the dataset being huge where the number of features exceeds the capabilities of WFS, vector tilesets showed extremely high performance in terms of visualizing datasets.

Besides interpreting the results the limitations of this research work should also be considered. Transmission of the data over the network is heavily dependent on the quality of the connection and it is obvious that not all Web GIS clients have access to the high-end Internet. However, all the benchmarks in this study have been conducted using high-speed network connection only. In the problem definition, it has been mentioned that another main factor of performance is hardware resources. During the benchmarks, the same specifications have been used and the changes in the hardware resources will have a considerable effect on the results. Lastly, in most realistic scenarios, more than one client will request access to the spatial data at the same time. However, during the tests in the methodology, no concurrency has been taken into account, and all the benchmarks were conducted via one client.

To sum up, from the benchmarks it has been revealed that when the dataset is voluminous and contains a couple of hundreds of thousands of features, pre-generated tilesets-based data access and visualization showed the best performance overall.

6. Conclusion

High-performance web map applications create a smooth user experience, increase usability, and therefore widen the audience of Web GIS applications. Measuring the performance of applications via the inclusion of benchmark analysis and tests is a crucial approach to generating indicators for the usability and user experience (Yang et al., 2011). Understanding the limitations of Web-based cartography, application and components, developers, cartographers, and scientists can collaborate on where improvements can be made in the approaches of Web mapping.

This research work focused on identifying data access and visualization methods of Web GIS which provides the highest possible performance and usability when the input is voluminous vector data.

The first research question aims to focus on the current state and challenges of Web GIS. In the problem definition section, several challenges have been introduced while the theoretical background section gave an overview of the underlying technologies, formats, and tools. For example, while interoperability is shown as one of the challenges in Web-based GIS applications, having common data formats across multiple tools such as XML and many globally-defined standards by organizations helps to mitigate this issue (Portele, 2007). Interoperability is normally achieved by defining the commonly accepted standards of the functional components, component interfaces and Web GIS elements (Bambacus et al., 2007).

The practical analysis has been conducted by employing 2 country-level large datasets and is dedicated to revealing the statistically meaningful results as much as possible. Before the benchmarks, several steps have been taken to prepare the spatial data infrastructure and the datasets; virtual machine instances have been created in the cloud environment, database and software tools installed, configured and both of the datasets have been imported into the database.

In the first part of the methodology OGC-defined common data access methods have been included in the tests and it has been discovered that for the datasets which are large and contain a couple of thousands of features these data access methods don’t provide the best performance and usability. It is also revealed that GeoJson as the vector data format showed a noticeable better performance compared to the XML. However, for the large vector datasets mentioned

above using a pre-generated vector tilesets-based approach showed the best performance to access the dataset over the web.

In terms of data visualization, tests showed that WebGL-based rendering via Mapbox GL JS has shown considerably better performance over the OpenLayers. As a generic method, vector tilesets-based data visualization showed the highest performance. During these tests possible performance variations due to the nature of the dataset such as different geometry types have also been considered by using layers with multiple geometry types such as lines and polygons.

To sum up, performing benchmarks under special conditions seems to be one of the most appropriate ways of choosing the best data access and visualization methods over the web while it has been obvious that considering the volume of the data is crucial. If the dataset is large where a couple of thousands of features have to be shown in the client application, ahead of time calculation and generation of the vector tiles is the best way to eliminate these above-mentioned issues.

Comparison of visualization and data access methods in a dynamic web mapping context for large geospatial vector datasets

Bakhtiyar Garashov

Summary

In recent years, web maps have transformed the way maps and geographic information are designed, produced, and distributed by cartographers (Cartwright, Gartner, Meng, & Peterson, 2010). Web GIS is a process of operating the traditional GIS on a web-based computing platform (Yang et al., 2005). It is difficult for WebGIS to perform well when large volumes of data and large numbers of concurrent users are involved (Cao, 2008). A major distribution method of spatial information these days is maps on the internet (Peterson, 2008). Therefore, it emphasizes the necessity of providing a reference for choosing the best data access and visualization approaches to the web environment.

The goal of this research was to evaluate prevalent data access and visualization methods for sharing and distributing big vector spatial datasets over the web. Several practical experiments were conducted and noticeable results have been made that it helps in the decision-making process when choosing an efficient method in terms of accessing and visualizing large geospatial vector datasets in the web-based GIS environments. In the problem definition section, the current state, challenges, and main problems have been discussed as it is the main objective of the first research question.

Two country-level datasets - Estonian Topographic Database and Estonian Soil classification datasets have been included to conduct the tests that have been introduced in the methodology of the study. The main reason behind the decision of choosing country-level datasets was to get the datasets as large as possible so that during the benchmarks it will be revealed how the performance of modern Web GIS applications would fluctuate due to the size of input data . As it has been mentioned in the problem definition section that hardware resources being used have an enormous effect on the performance of multiple software tools, systems, and libraries. Therefore, to mitigate any misleading effect of this all the tests have been conducted in the same type of hardware machines in the cloud environment.

In the first part of the methodology, a number of tests have been conducted to reveal the most efficient and suitable data access methods to distribute large geospatial vector datasets in the web

environment. In terms of data access methods, it has been revealed that using WFS via GeoJson as output showed better performance if the number of features is not exceeding a certain upper limit (~5000 features) while for the datasets where a couple of thousands of features is the case, vector tilesets-based approach showed the best results. Although for the whole dataset WMS showed better performance taking into account that WMS uses server-side rendering to deliver the map to the end-user as a picture and discards the advantages of vector data it can be concluded as a limitation. When comparing the XML and GeoJson formats as output for WFS it has been mentioned that in terms of both received data size and average response time GeoJson showed noticeable better performance due to the resource effective nature of this format. A similar result has been mentioned by Nurseitov et al., 2009. Furthermore, L. Li et al., 2017 described that GeoJson is resource-saving and human-readable whilst XML is well-known for its verbosity and ineffectiveness in terms of costs while storing and transmitting.

Moreover, taking into account the applicability of common computer science techniques in the Web GIS 2 caching solutions have been included to increase the performance of simple WMS requests. During both of the tests via multiple caching techniques it has been revealed that both tiling-based and web proxy caches help to reduce the average response time considerably. Quinn & Gahegan, 2010 describes the effectiveness of the tilesets caching in the Web GIS and Blower, 2010 described the caching from the perspective of static file caching. However, the last study is only limited to using one tool to cache raster images and no recent study has been found which reveals the usage of the web proxy caching method in the web cartography and the current study fills this gap.

In the data visualization methods, the study revealed that using Web GL-based web mapping libraries is a feasible option for rendering a big number of features, and using vector tilesets performed the best result. Before the benchmarks, a usability analysis has been performed to select the suitable libraries from multiple options. Overall, the pre-generated vector tilesets-based approach gives advanced and efficient possibilities to access and render voluminous vector spatial datasets over the web. To mitigate the transmission issue of large vector datasets it is important to send the data over the network in small-sized chunks (Shang, 2015) which is an approach used by vector tiles currently. Although Olasz et al., 2016 mentioned server-side

rendered data visualization in geographical context and provided analysis via conducting a survey the work only focused on generating and distributing raster data based tiling schema.

The novelty of this study is to include multiple scenarios of vector tilesets-based approach in the benchmarks such as pre-generated, dynamically generated, served statically, and using special tile server where previous studies mostly limited with only using pre-generated Mapbox vector tiles. The study has revealed the advancements in using vector tiles and provided a reference for future works in the same field with practical benchmark analysis.

Suuremahuliste vektor-ruumiandmete visualiseerimise ja juurdepääsu meetodite võrdlus dünaamiliste veebikaartide kontekstis

Bakhtiyar Garashov

Kokkuvõte

Veebikaardid on viimastel aastatel muutnud kaartide ja geograafilise informatsiooni kujundamise, loomise ja levitamise viise (Cartwright, Gartner, Meng, & Peterson, 2010). Veebi-GIS on süsteem, kus traditsiooniline GIS juhindub veebipõhisest arvutusplatvormist (Yang et al., 2005). Suurte andmemahtude või suure hulga samaaegsete kasutajate korral ilmnevad veebi-GISi kasutamisel või läbi selle päringute tegemisel jõudlusprobleemid (Cao, 2008). Veebikaardid on tänapäeval üks olulisemaid ruumilise informatsiooni levitamise viise (Peterson, 2008). See toob esile ruumiandmete veebipõhiste juurdepääsu ja visualiseerimise meetodite võrdlemise vajalikkuse.

Uurimistöö eesmärk on hinnata laialdases kasutuses olevate andmete juurdepääsu tööriistade ja visualiseerimise meetodite sobivust suuremahuliste vektor-andmekogude jagamiseks läbi veebi. Uurimistöö jooksul viidi läbi mitmeid eksperimente, mille tulemused aitavad teha valikut optimaalse suuremahulise veebipõhise vektorandmestiku juurdepääsu ja visualiseerimise meetodite osas. Probleemipüstituse osas on kirjeldatud hetkeolukord, väljautsed ja peamised probleemid, mis on esimese uurimisküsimuse peamiseks eesmärgiks.

Uurimistöös on kasutatud Eesti Topograafiline Andmekogu ja Eesti mullakaardi andmeid, et läbi viia metoodikaosas kirjeldatud testid. Lähteandmestikud on valitud võimalikud suuremahulised, et näha erinevusi testide tulemustes tänapäevaste veebi-GIS rakenduste jõudluses edastada ja visualiseerida erineva mahuga andmeid. Riistvaralised erinevused mõjutavad oluliselt mitmete tarkvarade, süsteemide ja teekide jõudlust. Seetõttu on kõik testid läbi viidud sama riistvaraga pilvekeskkonnas.

Metodoloogia esimeses osas viidi läbi mitmeid teste, et leida sobivaimad ja efektiivsemad andmete juurdepääsu pakkumise meetodid suuremahuliste vektor-ruumiandmete jagamiseks veebikeskkonnas. Andmete juurdepääsu meetoditest, mille nähtuste arv oli kuni 5000, näitas parimat jõudlust GeoJsonit edastav WFS-teenus, samas kui paari tuhande nähtusega andmestikus

oli parima jõudlusega vektor-ruutvõre. Kogu andmestiku korral näitas parimat jõudlust WMS-teenusel põhinev lahendus. Siiski seab piiranguid see, et WMS kasutab serveri-poolse kaardipildi renderdamist, et esitada lõppkasutajale kaart pildi kujul ja seeläbi heidab kõrvale mitmed vektorandmetele omased eelised. WFS väljundandmetena näitas GeoJson XML-st märgatavalt paremat jõudlust nii edastatud andmemahu kui keskmise edastusaja lõikes. Sama tulemuseni on jõudnud ka Nurseitov et al. (2019). Li et al. (2017) on väitnud, et GeoJson on ressursisäästlik ja inimesele lihtsamini mõistetav formaat kui XML, mis on teada-tuntud oma paljusõnalise süntaksi ja ebaefektiivsuse poolest andmete säilitamisel ja edastamisel.

Arvestades tavapäraseid arvutiteaduste tehnikaid Web GIS 2-s kaasati uurimistöösse erinevad andmete vahemällu salvestamise lahendused, et parandada WMS taotluste jõudlust. Nii võrepõhine kui puhverserveri (e. Proxy serveri) põhised vahemällu salvestamise tehnikad vähendavad märkimisväärselt päringu tegemiseks kuluvat aega. Quinn & Gahegan (2010) kirjeldavad ruutvõrede vahemällu talletamise efektiivsust veebi-GISis ja Blower (2010) kirjeldab vahemällu salvestamist staatilise faili perspektiivist. Samas on Blower (2010) uuringu piiranguks see, et on kasutatud ainult ühte tööriista, et talletada vahemällu rasterpilt ja ükski hiljutine uuring ei kirjelda puhverserverite meetodi rakendatavust veebikartograafias. Käesolev uurimistöö täidab seda lünka. Andmete visualiseerimise meetoditest pakkusid WebGL veebikaartide teegid võimalikku lahendust suure hulga nähtuste renderdamiseks ja vektor-ruutvõred omasid parimat jõudlust. Enne jõudlusnäitajate leidmist viidi läbi rakendatavusanalüüs, et leida kõigist valikutest sobivaimad teegid. Et vähendada suuremahuliste vektorandmete edastamise probleemi on oluline saata andmeid läbi võrgu väikesemahuliste osadena (Shang, 2015), mis on ka lähenemine, mida vektor-ruutvõre praegu kasutab. Kuigi Olasz et al. (2016) mainisid andmete geograafilist visualiseerimist läbi renderdamise ja viisid läbi selleteemalise küsitluse, keskendus nende töö üksnes raster-võrede loomisele ja jagamisele.

Käesoleva uurimistöö uudsus seisneb mitmete vektor-ruutvõredel põhinevate stsenaariumite, nagu eelgenereritud, dünaamiliselt genereeritud, staatiliselt edastatud ja spetsiaalsete võreserverite kasutamine, kaasamisel jõudlustestidesse. Varasemates uuringutes on enamasti piirdutud üksnes eelgenereeritud Mapbox vektor-ruutvõredega. Uurimistöö demonstreeris vektor-ruutvõrede edasiminekut ja on võrdlusaluseks edaspidistele samas valdkonnas jõudlusanalüüse tegevatele uurimistöödele.

Acknowledgements

I would like to thank my supervisor Alexander Kmoch for his cooperation and assistance. The guidance and support he provided throughout this research were invaluable; without them, it would have been impossible to complete the project.

My deep appreciation extends to all my teachers at the University of Tartu and more specifically, the academic staff of the Department of Geography, who has helped me become who I am today. Also, I am grateful to my managers and colleagues in the Mooncascade who have been always motivating and encouraging me throughout the period of my thesis.

Having the University of Tartu as my Alma Mater is an honor for me. Finally, I want to extend my gratitude to my parents, family and friends who have supported me through this process.

References

Adnan, M., Singleton, A., & Longley, P. (2010). *Developing efficient web-based GIS applications*. 16.

Alesheikh, A. A., Helali, H., & Behroz, H. (2002). Web GIS: technologies and its applications. In Symposium on geospatial theory, processing and applications (Vol. 15). Retrieved from https://www.researchgate.net/publication/228714608_Web_GIS_Technologies_and_its_applications

Agrawal, S., & Gupta, R. D. (2017). Web GIS and its architecture: A review. *Arabian Journal of Geosciences*, *10*, 518. https://doi.org/10.1007/s12517-017-3296-2

Alimuddin, A. A., Niswar, M., Amirullah, I., & Rijal, M. (2020). Design and Implementation of REST API for Academic Information System. *IOP Conference Series: Materials Science and Engineering*, *875*, 012047. https://doi.org/10.1088/1757-899X/875/1/012047

Antoniou, V., Morley, J., & Haklay, M. (2009). Tiled Vectors: A Method for Vector Transmission over the Web. In *Web and Wireless Geographical Information Systems* (Vol. 5886, p. 71). https://doi.org/10.1007/978-3-642-10601-9_5

Bambacus, M., Yang, C., Evans, J., Cole, M., Alameh, N. & Marley, S. (2007) ESG: An interoperable portal for prototyping applications. URISA Journal, 19 (2), 15–21.

Behr, F.-J., Holschuh, K., Wagner, D., & Zlotnikova, R. (2011). *Vector Data Formats in Internet based Geoservices*. https://doi.org/10.1201/b11080-27

Brinkhoff, T. (2007). Increasing the fitness of OGC-Compliant Web Map Services for the Web 2.0. In S. I. Fabrikant & M. Wachowicz (Eds.), *The European Information Society: Leading the Way with Geo-information, Proceedings of the 10th AGILE Conference, Aalborg, Denmark, 8-11 May 2007* (pp. 247–264). Springer. https://doi.org/10.1007/978-3-540-72385-1_15

Cartwright, W., Gartner, G., Meng, L., & Peterson, M. P. (2010). Lecture Notes in Geoinformation and Cartography. Retrieved from www.springer.com

Carpendale. (2003). Considering Visual Variables as a basis for Information Visualisation.175–188. Retrieved from https://prism.ucalgary.ca/bitstream/1880/45758/2/2001-693-16.pdf

Cao, Y. (2008) Utilizing grid computing to optimize real-time routing. George Mason University Ph.D. Dissertation. p. 107.

Coetzee, S., Ivánová, I., Mitasova, H., & Brovelli, M. A. (2020). Open Geospatial Software and Data: A Review of the Current State and A Perspective into the Future. *ISPRS International Journal of Geo-Information*, *9*(2), 90. https://doi.org/10.3390/ijgi9020090

Cohen-Almagor, R. (2011). Internet History. *International Journal of Technoethics*, *Vol. 2*, 45–64. https://doi.org/10.4018/jte.2011040104

Corner, S., (2010) The 8-second rule, Available on http://www.submitcorner.com/Guide /Bandwidth/001.shtml

Doyle, A. (n.d.). *OPENGIS PROJECT DOCUMENT 00-028*. 45.

Giuliani, G., Dubois, A., & Lacroix, P. (2013). Testing OGC Web Feature and Coverage Service performance: Towards efficient delivery of geospatial data. *Journal of Spatial Information Science*, *7*. https://doi.org/10.5311/JOSIS.2013.7.112

Goodchild, M. F., Steyaert, L. T., Parks, B. O., Johnston, C., Maidment, D., Crane, M., & Glendinning, S. (1996). *GIS and Environmental Modeling: Progress and Research Issues*. John Wiley & Sons.

Guo, H. (2017). Big Earth data: A new frontier in Earth and information sciences. *Big Earth Data*, *1*(1–2), 4–20. https://doi.org/10.1080/20964471.2017.1403062

Gaffuri, J. (2012). Toward web mapping with vector data. Geographic information science, 87–101.

Hearnshaw, H. M., Unwin, D., & Association for Geographic Information (Eds.). (1994). *Visualization in geographical information systems*. Wiley & Sons.

Internet World Stats 2022. https://www.internetworldstats.com/stats.htm Last retrieved on March, 20 2022

Jacksi, K., & Abass, S. M. (2019). *Development History Of The World Wide Web*. *8*(09), 5.

J. D. Blower. Gis in the cloud: Implementing a web map service on google app engine. In Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application, COM.Geo '10, pages 34:1–34:4, New York, NY, USA, 2010. ACM.

Kitchin, R., & McArdle, G. (2016). What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets. *Big Data & Society*, *3*(1), 2053951716631130. https://doi.org/10.1177/2053951716631130

Kmoch, A., Kanal, A., Astover, A., Kull, A., Virro, H., Helm, A., Pärtel, M., Ostonen, I., & Uuemaa, E. (2021). EstSoil-EH: A high-resolution eco-hydrological modeling parameters dataset for Estonia. *Earth System Science Data*, *13*(1), 83–97. https://doi.org/10.5194/essd-13-83-2021

Kraak, M. J. (2001). Settings and needs for web cartography. *Web Cartography : Developments and Prospects*, 1–8.

Lee, J.-G., & Minseo, K. (2015). Geospatial Big Data: Challenges and Opportunities. *Big Data Research*, *2*. https://doi.org/10.1016/j.bdr.2015.01.003

Leiner, B. M., Kahn, R. E., & Postel, J. (2009). A Brief History of the Internet. *ACM SIGCOMM Computer Communication Review*, *39*(5), 10.

Li, S., Dragicevic, S., & Veenendaal, B. (2011). *Advances in web-based GIS, mapping services and applications*. CRC Press/Balkema. https://doi.org/10.1201/b11080

Li, L., Hu, W., Zhu, H., Li, Y., & Zhang, H. (2017). Tiled vector data model for the geographical features of symbolized maps. PloS one, 12(5), e0176387.

Lupp, M. (2008). OGC Web Services. In S. Shekhar & H. Xiong (Eds.), *Encyclopedia of GIS* (pp. 799–800). Springer US. https://doi.org/10.1007/978-0-387-35973-1_903

McGarty, T. (2002). *The Internet Protocol (IP) and Global Telecommunications Transformation*.

Michaelis, C., & Ames, D. (2017). *Web Feature Service (WFS) and Web Map Service (WMS)*. https://doi.org/10.1007/978-3-319-17885-1_1480

Morris, S. (2006). Geospatial Web Services and Geoarchiving: New Opportunities and Challenges in Geographic Information Service. *Library Trends*, *55*. https://doi.org/10.1353/lib.2006.0059

Mozilla developer network. (2022). *JSON - JavaScript | MDN*. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON. Last retrieved on 17 Mar, 2022.

Netek, R., Masopust, J., Pavlicek, F., & Pechanec, V. (2020). Performance Testing on Vector vs. Raster Map Tiles—Comparative Study on Load Metrics. *ISPRS International Journal of Geo-Information*, *9*(2), 101. https://doi.org/10.3390/ijgi9020101

Neumann, A. (2008). Web mapping and web cartography. In Encyclopedia of GIS (pp. 1261–1269). Springer.

Nurseitov, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009). Comparison of JSON and XML Data Interchange Formats: A Case Study. *Undefined*. https://www.semanticscholar.org/paper/Comparison-of-JSON-and-XML-Data-Interchange-A-Case-Nurseitov-Paulson/84321e662b24363e032d680901627aa1bfd6088f

Olasz, A., Nguyen Thai, B., & Kristóf, D.. A new initiative for tiling, stitching and processing geospatial big data in distributed computing environments. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 3:111, 2016.

Open GeoSpatial Consortium E-learning platform. Last modified on 21 Feb, 2022. Last retrieved on 18 Mar, 2022. http://opengeospatial.github.io/e-learning/index.html

Openstreetmap contributors 2015: Slippy map tile names. https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames Last retrieved on 20 Mar, 2022.

Open Geospatial Consortium, 2006b. “OpenGIS Implementation Specification for Geographic Information – Simple Feature Access - Part 2: SQL option”.Available at http://www.opengeospatial.org.

Peterson, M. (2003). *Maps and the Internet* (pp. 1–16). https://doi.org/10.1016/B978-008044201-3/50003-7

Peterson, M. P. (2008). International perspectives on maps and the Internet. Heidelberg Berlin: Springer.

Pullar, D., Torpie, D., & Barker, T. (2011). *Building web services for public sector information and the geospatial web* (pp. 109–118). https://doi.org/10.1201/b11080-10

Pundt, H. (2000). Visualization of spatial data for field based GIS. *Computers & Geosciences*, 51–56.

Portele, C. (2007) OpenGIS Geography Markup Language (GML) encoding standard. [Online] Available from: http://portal.opengeospatial.org/files/index.php?artifact_id=20509&passcode=p74xjvt23hm328389rks Last accessed on 14th April 2022.

Qian, Z., Wang, P., Zhang, L., & Yang, C. (2004). *OpenGIS WMS implementation and its integrated application using ASP.NET*. 2953–2956 vol.5. https://doi.org/10.1109/IGARSS.2004.1370314

Quinn, S., & Gahegan, M. (2010). A predictive model for frequently viewed tiles in a web map. Transactions in GIS, 14(2), 193–216.

Ramsey, P. (2013). *Refractions Research: Mashing up the Enterprise Article*. http://www.refractions.net/expertise/whitepapers/mashups/mashups/

Reichardt, M., & Robida, F. (2019). Why Standards Matter – The objectives and roadmap of the International Open Geospatial Consortium (OGC). *Annales Des Mines - Responsabilite et Environnement*, *94*(2), 25–29.

Roth, R. (2013). Interactive Maps: What we know and what we need to know. *Journal of Spatial Information Science*, *6*, 59–115. https://doi.org/10.5311/JOSIS.2013.6.105

Roth, R. E., Donohue, R. G., Sack, C. M., Wallace, T. R., & Buckingham, T. M. A. (2014). A Process for Keeping Pace with Evolving Web Mapping Technologies. *Cartographic Perspectives*, *78*, 25–52. https://doi.org/10.14714/CP78.1273

Roth, R. E., Robinson, A., Stryker, M., Maceachren, A. M., Lengerich, E. J., & Koua, E. (2008). Web-based geovisualization and geocollaboration: Applications to public health. *In Proceedings of Joint Statistical Meeting*.

Schaub, T., Doyle, A., Daly, M., Gillies, S., & Turner, A. (2006). *GeoJSON draft version 6—GeoJSON*. http://wiki.geojson.org/GeoJSON_draft_version_6

Shanzhen, Y., Zhou, L., Xing, C., Qilun, L., & Zhang, Y. (2001). Semantic and interoperable WebGIS. *Proceedings of the Second International Conference on Web Information Systems Engineering*. https://doi.org/10.1109/WISE.2001.996702

Shang, X. (2015). A Study on Efficient Vector Mapping With Vector Tiles Based on Cloud Server Architecture (Unpublished doctoral dissertation). University of Calgary.

Shekhar, S., Evans, M. R., Gunturi, V., Yang, K., & Cugler, D. C. (2014). Benchmarking Spatial Big Data. In T. Rabl, M. Poess, C. Baru, & H.-A. Jacobsen (Eds.), *Specifying Big Data Benchmarks* (pp. 81–93). Springer. https://doi.org/10.1007/978-3-642-53974-9_8

Shekhar, S., Raju Vatsavai, R., Sahay, N., Burk, T. E., & Lime, S. (2001). WMS and GML based interoperable web mapping system: ACM-GIS 2001: Proceedings of the Ninth ACM International Symposium on: Advances in Geographic Information Systems. *Proceedings of the ACM Workshop on Advances in Geographic Information Systems*, 106–111.

Siddiqa, A., Hashem, I., Yaqoob, I., Marjani, M., Shamshirband, S., Gani, A., & Nasaruddin, F. (2016). A Survey of Big Data Management: Taxonomy and State-of-the-Art. *Journal of Network and Computer Applications*, *71*. https://doi.org/10.1016/j.jnca.2016.04.008

Soomro, T. R., Zheng, K., & Pan, Y. (1999). HTML and multimedia Web GIS. *Proceedings Third International Conference on Computational Intelligence and Multimedia Applications. ICCIMA'99 (Cat. No.PR00300)*. https://www.academia.edu/6831905/HTML_and_multimedia_Web_GIS

Tiwari, A., & Jain, K. (2013, November 21). *Geospatial Framework For Dengue using Open Source Web GIS Technology*.

Wong, S. H., Swartz, S. L., & Sarkar, D. (2002). A Middleware Architecture for Open and Interoperable GISs. *IEEE MultiMedia*, *9*(2), 62–76. https://doi.org/10.1109/93.998065

Yang, C., Raskin, R., Goodchild, M., & Gahegan, M. (2010). Geospatial Cyberinfrastructure: Past, present and future. *Computers, Environment and Urban Systems*, *34*(4), 264–277. https://doi.org/10.1016/j.compenvurbsys.2010.04.001

Yang, B., Purves, R., & Weibel, R. (2007). Efficient transmission of vector data over the Internet. Retrieved from www.tandfonline.com

Yang, C., Wong, D., Yang, R., Kafatos, M. & Li Q. (2005) Performance improving techniques in WebGIS. International Journal of Geographical Information Science, 19 (3), 319–342.

Yao, X., & Li, G. (2018). Big spatial vector data management: A review. *Big Earth Data*, *2*(1), 108–129. https://doi.org/10.1080/20964471.2018.1432115

Yang, C., Wu, H., Huang, Q., Li, Z., Li, J., Li, W., Miao, L., & Sun, M. (2011). WebGIS performance issues and solutions. In Advances in Web-based GIS, Mapping Services and Applications (pp. 121–138). CRC Press. https://doi.org/10.1201/b11080-12

Zhao, L., Chen, L., Ranjan, R., Choo, K.-K. R., & He, J. (2015). Geographical information system parallelization for spatial big data processing: A review. *Cluster Computing*, *19*(1), 139–152. https://doi.org/10.1007/s10586-015-0512-2

Annexes

Annex 1: GML format

An example of GML data structure from Estonian Topographic Dataset which contains an object in Point geometry type, properties and corresponding values.

```
▼<gml:featureMember>
 ▼<Msc-thesis:e_305_puittaimestik_p fid="e_305_puittaimestik_p.1">
    <Msc-thesis:etak_id>1623957</Msc-thesis:etak_id>
    <Msc-thesis:kood>305</Msc-thesis:kood>
    <Msc-thesis:kood_t>Puittaimestik</Msc-thesis:kood_t>
    <Msc-thesis:tyyp>60</Msc-thesis:tyyp>
    <Msc-thesis:tyyp_t>Harvik</Msc-thesis:tyyp_t>
    <Msc-thesis:vajalik>30</Msc-thesis:vajalik>
    <Msc-thesis:vajalik_t>Korras</Msc-thesis:vajalik_t>
    <Msc-thesis:andmeallika_id>214</Msc-thesis:andmeallika_id>
    <Msc-thesis:korgusallika_id>214</Msc-thesis:korgusallika_id>
    <Msc-thesis:ruumikujuallika_id>214</Msc-thesis:ruumikujuallika_id>
    <Msc-thesis:muutmisaeg>2019-05-23</Msc-thesis:muutmisaeg>
    <Msc-thesis:geom_muutmisaeg>2019-05-23</Msc-thesis:geom_muutmisaeg>
    <Msc-thesis:valjavote>2022-03-19</Msc-thesis:valjavote>
   ▼<Msc-thesis:wkb_geometry>
    ▼<gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#3301">
       <gml:coordinates xmlns:gml="http://www.opengis.net/gml" decimal="." cs="," ts=" ">665965.965,6516966.689,42.305</gml:coordinates>
     </gml:Point>
    </Msc-thesis:wkb_geometry>
  </Msc-thesis:e_305_puittaimestik_p>
</gml:featureMember>
```

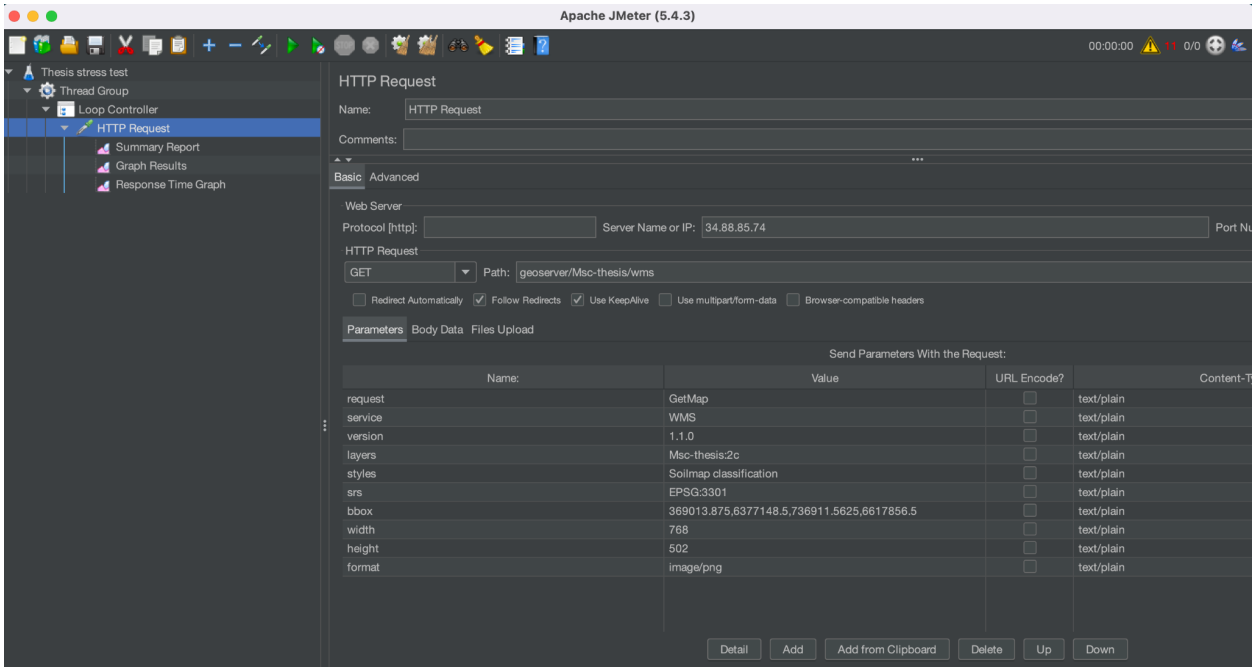
Annex 2: KML format

An example of a KML data structure

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <kml xmlns="http://www.opengis.net/kml/2.2" xmlns:ns2="http://www.google.com/kml/ext/2.2">
3      <Document>
4          <LookAt>
5              <longitude>-103.75528308474964</longitude>
6              <latitude>44.43280765338136</latitude>
7              <altitude>27628.008612851434</altitude>
8              <heading>0.0</heading>
9              <tilt>0.0</tilt>
10             <range>22324.88009438314</range>
11             <altitudeMode>clampToGround</altitudeMode>
12         </LookAt>
13     </Document>
14 </kml>
```

Annex 3: Apache JMeter configuration

Apache JMeter configuration for the WMS load test



Non-exclusive license to reproduce thesis and make thesis public

I, Bakhtiyar Garashov,

1. herewith grant the University of Tartu a free permit (non-exclusive license) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,
“Comparison of visualization and data access methods in a dynamic web mapping context for large geospatial vector datasets” supervised by Phd, Alexander Kmoch.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons license CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive license does not infringe other persons’ intellectual property rights or rights arising from the personal data protection legislation.

Bakhtiyar Garashov
Tartu, 28.05.2022