Rethinking the Usage and Experience of Clustering Markers in Web Mapping

Loïc Fürhoff[1]

[1]**University of Applied Sciences Western Switzerland (HES-SO), School of Management and Engineering Vaud (HEIG-VD), Media Engineering Institute (MEI), Yverdon-les-Bains, Switzerland**

Corresponding author:
Loïc Fürhoff[1]

Email address: loic.furhoff@heig-vd.ch

ABSTRACT

Although the notion of 'too many markers' have been mentioned in several research, in practice, displaying hundreds of Points of Interests (POI) on a web map in two dimensions with an acceptable usability remains a real challenge. Web practitioners often make an excessive use of clustering aggregation to overcome performance bottlenecks without successfully resolving issues of perceived performance. This paper tries to bring a broad awareness by identifying sample issues which describe a general reality of clustering, and provide a pragmatic survey of potential technologies optimisations. At the end, we discuss the usage of technologies and the lack of documented client-server workflows, along with the need to enlarge our vision of the various clutter reduction methods.

INTRODUCTION

Assuredly, Big Data is the key word for the 21st century. Few people realise that a substantial part of these massive quantities of data are categorised as geospatial (Lee & Kang, 2015). Nowadays, satellite-based radionavigation systems (GPS, GLONASS, Galileo, etc.) are omnipresent on all types of new digital devices. Trends like GPS Drawing, location-based game (e.g. Pokémon Go and Harry Potter Wizard Unite), Recreational Drone, or Snapchat 'GeoFilter' Feature have appeared in the last five years – highlighting each time a new way to consume location. Moreover, scientists and companies have built outstanding web frameworks and tools like KeplerGL to exploit geospatial data.

Nonetheless even today, the simplest case of displaying hundreds of Points of Interests (POI) on a web map in two dimensions with an acceptable usability can be a real challenge. Especially with the perspective of the user, the navigation experience can be utterly painful. Unquestionably, the choice of technology and their performance has a strong impact on the interactive experience as much as the choice of data visualisation. Furthermore, practitioners often only have a unique point of view due to their education limited to a single or a few number of fields. On the contrary, the discipline of geovisualisation embraces and requires knowledge in many fields: Mathematics, Geoinformatics, Geospatial, Web Development, Accessibility, Information Architecture, Data-visualisation, etc.

The present paper tries to bring about a general awareness by presenting and discussing the use of different technologies and their effects on common performance bottlenecks. The ambition is to provide a modest and pragmatic survey of the approaches to enhance web map user experience with a focus on the perceived performance and question practices of today.

THE PRACTICE OF AGGREGATION

The modern web map history certainly began with the launch of Google Maps in 2005. Even though MapQuest and Yahoo companies made an appearance a few months before, the public paid attention to this new kind of medium only a bit later (Gibbs, 2015). Arranging a large quantity of markers on a web map rapidly land on the issue list. Therefore basic aggregation libraries using clustering methods were published during the same year (Poskanzer (2005) is one of the pioneers).

The usage of aggregation originated from the necessity to overcome mainly two points. At that time, the remarkable Google maps Pin made its way along with the map technology. The pin certainly contributes to the popularity of the service and became a popular representation to the point of being iconic and to be acquired almost ten years later by the Museum of Modern Art (MoMa) in their permanent collection (Rasmussen Eilstrup, 2014). The unique shape of this icon allows the user to locate a precise place without obscuring the area. Regardless this peculiar design, some characteristics like the vertical shape and the drop shadow also emphasised a classic problematic – the cluttering of icons. The visualisation system is confronted by the limitation of preserving markers with a reasonable and accessible size which allow the user to see them and click on them, meanwhile letting the user to interact with the zoom property of the map. Markers, even with few of them, can overlap themselves at the minimum zoom level. A large number of points reduce proportionally the potential usable minimum zoom level. A lot of research has been made on finding algorithms for automatic symbol placement such as Point-Feature Label Placement(PFLP) or for instance the conflict graph algorithm. However markers have specific attributes which render these mechanisms inoperative (Burigat & Chittaro, 2008). Unlike labels, POIs cannot simply be removed from the map without clearing away important information when the available space is insufficient. In contrary, the existence of markers on a map in most cases is to highlight information. Markers also differentiate from labels by using a fixed location without conceding flexibility.

The second point of using aggregation is the proportional degradation of performance for each marker appended to a map. In web application, performance covers a large range of topics explained by the inherent complexity of the client-server model at the core of the World Wide Web. Displaying markers inside a Geographic Information System (GIS) software, as it was before web map or today to prepare and review a data set, bypass this complexity. The main one being that the data and the GIS software are on the same computer. Whereas on the web, vulnerabilities or weak points can arise at multiple layers of the network model.

Visualise aggregation

In practice for visualising an abundance of markers and in the field of interactive web map, only a couple of visualisations are suitable: Heatmaps, Tiled Heatmaps (grid of rectangles, hexagons, etc.) and Markers Clustering.

Meier (2016) presented, compared and discussed these methods. Delort (2010) tried to use Voronoi visualisation as another alternative. Other representations exist, nonetheless they are generally adopted in the field of statistics which allow the system to have a limited interactivity of the zoom feature and allow the marker to be striped of its intrinsic attributes like the fixed position.

Without being able to quantify it, developers intuitively seem to choose clustering methods more often when they need a visualisation to be understood by many. Also according to the survey of Meier (2016) and in terms of user comprehension, the clustering of markers is a method better understood for density of precise spatial location or point data like the location of a restaurant or a hotel whereas (tiled) heatmaps would be more suited to areal data like the population density of a country. The aim of this paper is to concentrate on the representation of precise and fixed geographical point data, therefore we will focus on clustering methods.

PERCEIVED PERFORMANCE ISSUES

Paradoxically since a few years, performance as a topic is making a huge comeback in web development. Each year, mobile devices receive superior processors and faster electronics, but Web-giant company like Google are still pushing for better performance (Osmani, 2018) – perhaps to satisfy their thirst of expansion for new markets and users. In the meantime, several studies showed the relation between the Time to Interactive (TTI) measure, in other words, the time that a user is waiting before being able to use a web page, and the percentage of users quitting a web page or on the contrary the increase of conversion when a low TTI is measured (Alstad, 2015). The result is a long list of new Human-centric metrics and application programming interface (API) to assess client-side latency and the perceived performance including the Performance Timeline API, the Navigation Timing API, the User Timing API, and the Resource Timing API (Wagner, 2017).

Around four years after the launch of Google Maps, multiple JavaScript libraries were present on the market to defeat overlapping issues with aggregation. Svennerberg (2009) established one of the first

comparisons of these algorithms to focus on performance. He tries to measure the total load time of each library towards the main popular web browsers. This kind of TTI analysis (with the tool of the time), like many others, considers only a subset of the performance issue – by examining the time for markers to be transmitted and rendered by the clustering algorithms. However many other human factors step in which are sometimes not easy to evaluate and compare with numbers and metrics. In this section, we'll try to list some of the common current and perceived issues with too many markers and clustering, impacting user experience beyond the aspect of loading times or classic lags. This following list represents only a fraction of the day-to-day issues bore by users with the purpose to point out a general reality.

Zoom Latency In the example Figure 1, the process, which happens after each interaction, is blocking all the layers (the markers and the base layer) and the navigation. This process affects the navigation and is generally caused by the loading of markers or the calculation of clusters.

Scattering Latency It happens that when a cluster is decomposed the script has a too high latency on recalculation which impacts the underlying animation (Fig. 2). Markers translate to their respective position to show their affiliations to the cluster. The experience is perceived as a non-necessary waiting time.

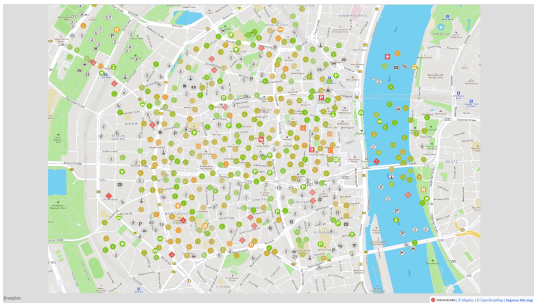


Figure 1. Video example of Zoom Latency
https://doi.org/10.6084/m9.figshare.9332690
https://wheelmap.org, Solzialhelden — © Mapbox — © OpenStreetMap

Figure 2. Video Example of Scattering Latency
https://doi.org/10.6084/m9.figshare.9332543
https://wheelmap.org, Solzialhelden — © Mapbox — © OpenStreetMap

Markers Flashing When the user interacts with the zoom, markers or clusters are disappearing by flashing or flickering (Fig. 3). Sometimes artefacts of old markers are appearing as well on the map.

Rendering Latency Markers size has a delay and does not follow the zoom interaction (Fig. 4). Besides the loss of comparison across zoom, the result is disturbing and specially not enjoyable.

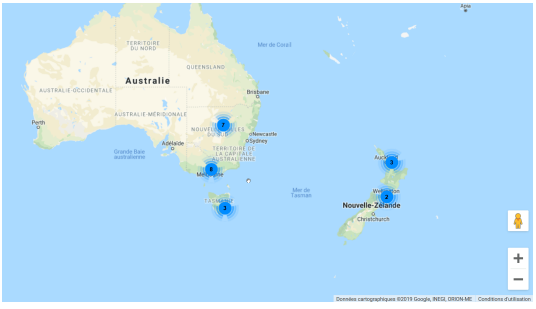


Figure 3. Video example of Markers Flashing with Google MarkerClusterer library
https://doi.org/10.6084/m9.figshare.9332237
https://developers.google.com/maps/documentation/javascript/marker-clustering, Map data © 2019 Google

Figure 4. Video example of Rendering Latency
https://doi.org/10.6084/m9.figshare.9332480
https://query.wikidata.org, Leaflet — Wikimedia — © OpenStreetMap contributors

Inadequate Gap Distance Chosen parameters or clustering method create too much distance between markers inside a cluster. Sometimes, developers choose to have a greater number of clusters to improve client-side latency (Fig. 5. However it also impacts the user experience. The user loses to some extent the perception of density.

Triggering Delay Having too many markers on a map sometimes constrain developers to make design choice. In Figure 6, the library is loading new markers only when the users release the click which creates a delay and reduce the smoothness of the experience.

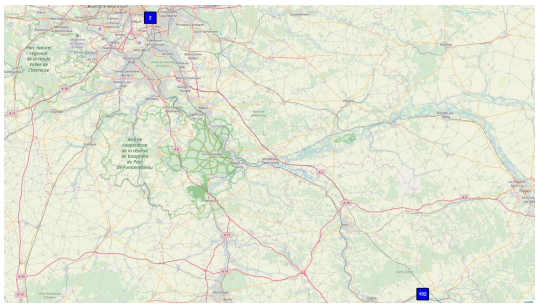


Figure 5. Video example of Inadequate Gap Distance
https://doi.org/10.6084/m9.figshare.9332105
https://bl.skywatch.ch/map, Leaflet — © OpenStreetMap

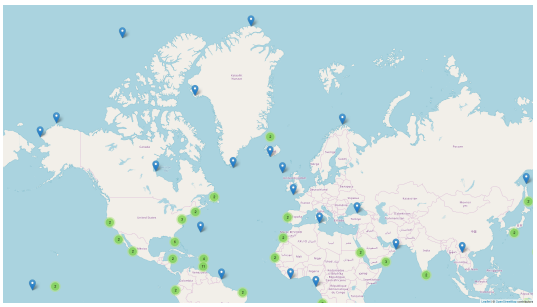


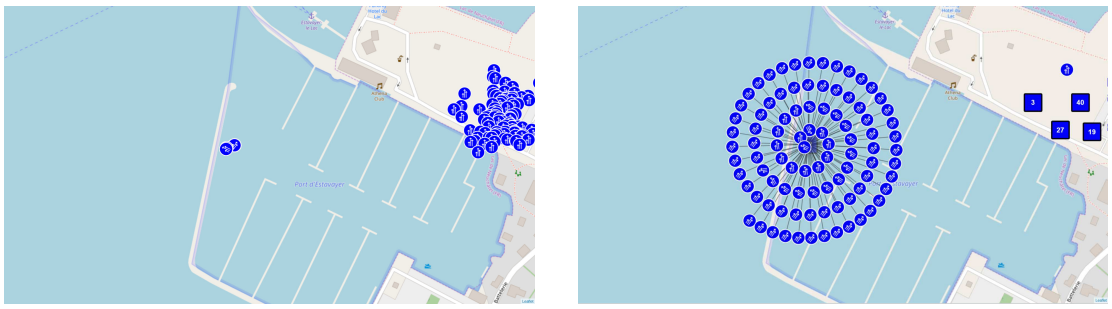
Figure 6. Video example of Triggering Delay
https://doi.org/10.6084/m9.figshare.9332588
https://github.com/mapbox/supercluster, Leaflet — © OpenStreetMap contributors

SPECIFIC POINTS TO CONSIDER

Before entering the main part of this paper, the readers should be aware that the results need to be only considered for a general usage. The following points are describing two situations where further observations are necessary.

Markers at the Same Exact Position

In some situation, it can happen that markers are positioned at the exact same place or within a too small distance. Such a distance is dependent on the maximum zoom level that a map allows or, in other words, the minimum scale of the map.



(a) Without spiderify method - only few markers are visible

(b) With spiderify method applied

Figure 7. 92 overlapping markers at the exact same position
https://bl.skywatch.ch/map, Leaflet — © OpenStreetMap

In the example of Skywatch BL (https://bl.skywatch.ch/map), the map was made to display points in time recorded from a mobile device which provide weather information such as temperature, UV index or humidity. The device is a niche product, and the development team choose to display a maximum of points to give a crowded impression. The User Interface (UI) allows the user to show points from a period

of times between one hour to one year. In Figure 7a, the screenshot show and edge case. The owner of a Skywatch BL device recorded points in a boat still attached to the dock.

To overhaul this problem, the unique practical way to do it nowadays is to 'spiderify' it. Instagram also found in the past another way to display overlapping localised photos using multiple icons to show a pile of photos (Newton, 2016). When the user clicked on a pile, the photos were shown in a grid (video example of the interaction : Door, 2012). However, it works only in the precise case of Instagram and their mobile application. In Figure 7b, the spiderify method grab all the markers placed in the same position and arrange them in a spiral. A straight line connects each marker to their real position. In practice, this method can be easily combined with clustering libraries.

This edge case originate from design issues. By all means, supposing that the control user interface only gave the possibility to select a precise date and not a period, the number of markers at the same position would be restrained, though the control would be certainly ineffective. In this case, the design team behind the map would need to evaluate and reconsider the usage of the map by questioning its purpose. e.g. Who are the users of the map? Is it a map made for contributors and therefore it needs to show every single point collected? Can we average the data? What are the type of the collected data between point and area data? etc.

Spatiotemporal Data

The use of spatiotemporal data, i.e point data which change overtime, can create particular issues on performance. Generally in those kinds of situations, the markers position are often fixed through time and have a shape which can fluctuate to reflect an indication of a state. The performance could be altered due to the update of markers. Indeed, standard library will simply renew all the markers each time they are adjusted. This specific issue will not be handled in this paper. For further information and an example of this problematic, please refer to Urbica (2018).

MULTIPLE LEVELS OF OPTIMISATIONS

In this section we will examine the anatomy of a web map and try to identify the mainsprings to enhance each level or layer. The objective is to get a simple and accessible overview of current and future web technology without deep and too technical knowledge. The study of Huang and Gartner (2012) asserted that there is a direct relationship between the increase of markers and the 'transmission load'. However, we'll see that we can greatly moderate this assumption and that we can exploit the client-server architecture at our advantages.

The first approach to battle with an abundance of points on a map is to try to diminish them based on their content. Either the markers are filtered in pre-processing stage or within the user interface. Chapter 3 of Huang and Gartner (2012) based on the classification of Recommender Systems (RSs) is already entirely devoted to this technique. The aim of this present paper is to examine purely technical methods therefore it is assumed that readers have taken cognisance of content-based optimisations like Huang and Gartner (2012) to filter markers which should be the first consideration in a 'too many markers' problem.

File size and decoding time

This subsection aims at improving principally loading time and transmission issues that Svennerberg (2009) early alluded.

Without doubt, GeoJSON, the standardised extension of the JSON, has become de facto standard format in web mapping. This versatile format supports multiple types of geometries and every common library is supporting it. The format accepts the extension of the data with new attributes which in our case is essential to link marker position with a describing content and then display supplementary data with an overlay for example.

One of the major reasons behind the adoption of GeoJSON is the tidy and human-readable appearance of the data opposed to the verbosity of the venerable XML format. The notion of human readable is generally in opposition to the consideration of transmission over a network. Consequently many advise and attempts to optimise have been enunciated and described for a long time in the geospatial community (Sandvik, 2012; Tihonov, 2014) including removing unused properties, reducing precision of coordinates, 'minifying' (removing all unnecessary characters for instance whitespaces or new line characters), applying Delta and zigzag encoding on coordinates.

The common belief and potential error for novices is to shrink the file size as far as possible, howbeit optimising file size may impact decoding time and can be counterproductive. In that perspective, proposed extensions of GeoJSON made some waves recently: GeoJSONL, Newline Delimited JSON (ndjson), JSON Lines (jsonl), and GeoJSON Text Sequences. All share the same objective to improve parsing by merely reorganising the structure of the GeoJSON. The file size stays the same as the original file which has been optimised with other techniques or not, but the memory usage to decode the file is greatly decreased which results in a lower latency in our web browser usage.

Binary File Format

Despite all the research, file size optimisation has its limits. Considering the display of thousands of markers in a web map, it's not rare to have to serve a GeoJSON of several tens of megabytes even after optimisation or using techniques like lossless server compression with GZIP, Zopfli or Brotli algorithms. GeoJSON was principally made as an interchange file format and not designed for transmission. The general suggested alternative to JSON is the use of a binary encoding format which in theory should be lighter and faster at parsing. Protocol Buffers (Protobuf) developed by Google is the most widespread serialisation library and by extension format. Two major organisations have adopted this library for their own usage. The Openstreetmap association and the company Mapbox are using it for storing and serving vector data whilst having totally different specification. In practice, Protobuf seems to deliver mixed results (Bernstein, 2014; Wen, 2017; Krebs, 2017; Larsgård, 2017), though it has a clear advantage in compressing numbers which probably explain the choice of Mapbox to use it in their vector tile specification. In fact, lossless server compression like GZIP as a great impact on GeoJSON and compare favourably with Protobuf. It's not certain that JSON and GeoJSON will be replaced in the future with such a solution. Although Protobuf is largely supported by Google on multiple programming language and stacks, it requires a complete other infrastructure and paradigm. In fact, the first intention of Google is to package Protobuf within their gRPC communication architecture. Saw by many an alternative to REST and GraphQL, gRPC is not principally directed at back-ends for communication between a server and a web browser but rather for native applications or low-powered appliances which require high-speed transmission or persistent connection e.g. IoT devices.

Computation

Algorithms and calculations are playing a large part in issues like zoom (Fig. 1) and scattering latency (Fig. 2). In those situations, markers are generally already loaded and cached which means that the calculation of clusters should be accounted for these issues. Some optimised systems are requesting only markers present inside the browser viewport and in that case, each zoom needs another network request. However even for those systems, the interaction should be smoother with caching strategies at least when the zoom level is reached a second time, but it is often not the case. In this section, we'll first examine how we can alleviate the calculation of clusters and try to get an elementary view on clustering algorithms.

Relieve the UI Thread

Some libraries are implementing solution to soften the perception of latency. The approach involves using different loading techniques. The first noticeable one is chunk loading. With a basic algorithm, the data are cut according to time of loading. This workaround allows the browser to alternate between the loading of the markers data and other elements or layers of the web map. The noticeable gain is that lags are shorter and the perception of speed slightly improved, but it does not completely solve the problem.

An alternative would be to push all or the main computing operations in parallel, in the background, with the Web Workers API. A part of frustrations are assuredly eliminated with just this method since the navigation will not be blocked, but the user will be disoriented as the loading will appear asynchronous.

Just as well as a binary format are invading the browser, the development of compiled programming language is slowly appearing on the browser side. WebAssembly (Wasm) is an open standard promoted by the World Wide Web Consortium (W3C). It allows to use low-level language like C/C++, D or Rust and have near-native performance which means lower time of calculation. Wasm was designed to work alongside JavaScript. However it's hard to tell how both will cohabit in the future. Although the technology is shipped in all the major browsers since November 2017 (McConnell, 2017), Wasm is still considered as experimental in the community and as far as we know, no tangible experience was done in web mapping as of today's date. This technology will need time to be propagated first, but mainly to take

down some essential limitations like the weakness of intermediate communications between Wasm and the JavaScript APIs which are essential, but affect the theoretical advantage of performance.

Clustering Algorithms

According to Wikipedia "Cluster analysis" (2019), the definition of cluster is very wide and was first described at least ninety years ago. The notion is used in multiple scientific disciplines. Consequently a numberless of algorithms were designed to answer the problematic of grouping data with notably different performances. Meert, Tronçon, and Janssens (2006) organised and vulgarised the major algorithms used in map application:

- K-means
- DBscan
- Hierarchical Clustering

Conceived in 1967, the first is the most common algorithm in clustering analysis due to its relative simplicity and performance. The second is relatively new (1996) in comparison and some research found that it is less efficient (Chakraborty, Nagwani, & Dey, 2014). However, DBscan has a lot of advantages: it does not require either a parameter indicating the number of clusters, or that points are within the same shape. DBscan is more flexible and allow arbitrarily shaped clusters (Wikipedia "DBSCAN", 2019). Nevertheless these characteristic are not essential to markers clustering in web mapping. Finally, hierarchical clustering make use of the prominent tree structure largely used in geospatial. In practice inside the clustering web map 'community', it was popularised by Dave Leaver on the plugin Markercluster for the Leaflet library and seems to be the most efficient (Agafonkin, 2016). Experiments have been made with other algorithms like sweep and prune which determine clusters by simulating collisions between markers. Grid-based clustering methods were also mentioned in an earlier version of the Google documentation (Mahe & Broadfoot, 2010) which explains how to handle many markers with Google Maps API. That said, the performance benefits were not really demonstrated for the sweep and prune method, and grid-based clustering definitely lack of meaning.

Spatial Indexes

The launch of the 'supercluster' library (Agafonkin, 2016) illustrated that pre-calculating spatial indexes in addition to a good algorithm has a considerable repercussion. Spatial indexes are at the core of spatial database, but tend to be useful to optimise also on the client-side. Without it, a database would need to sequentially compare every record to respond to a search query. Spatial indexes instead build a tree to retrieve the data at the fastest speed. Building an index certainly impact in some way the loading time which is precious in our case, but this initial cost is valuable later when used in conjunction with a clustering algorithm (Agafonkin, 2017). Supercluster, which manages to handle millions of points, use R-tree, but at least fifteen potential algorithms are known and use in the field (Wikipedia "Spatial database: Spatial index", 2019). There is always room for progression, but it's more unlikely that the future of optimisation in clustering markers will be determined by the spatial index and clustering algorithm to such an extent of what we went through with supercluster.

Client-Server Load Distribution

The typical implementation of clustering does not involve the server except for serving the data files. Optimisation of the data transmission was for a long period restrained to handle the lowest number of files. Each new file involves a round of costing Hypertext Transfer Protocol (HTTP) requests. Developers had to balance between the cost-benefit of the requests compared to the sizes of file. Icons sprites is assuredly the culminating example method of this period. Sprites targeted especially icons due to their small sizes. Icons were combined together to constitute a single file and a single transfer over the network. The icons were then separated by a masking mechanism. Since the end of 2015 (Schoors & Deveria, n.d.), major web browsers support version 2 of HTTP which completely change this paradigm by empowering concurrent transfers without degradation. The practice of chunks slicing emerged in multiple places. At the moment, the good practice in modern web development is to deliver the minimum piece of code only related to the current displayed page to minimise transmission and parsing time (Wagner & Osmani, 2019). This innovation will also certainly benefit long-established technology like Web Map Service (WMS).

Streaming

In reality before version 2 of HTTP, it was already possible to avoid the cost of multiple requests by using Chunked Transfer Encoding function of HTTP version 1.1 . Plug-in combined with libraries which support the streaming of GeoJSON exists. However in our case, this technique will only improve the First Meaningful Paint (the time it takes for an essential content to appear on the page). Markers will appear one after another and the user will still have to wait to have the big picture. The user could also be more confused without a clear feedback indicating the completion of loading. Streaming should be reserved to data which change over time with a short lifespan.

Preloading and Prefetching

As mentioned previously, zoom and panning can induce markers flashing due to computation or loading time. Recently, web browsers implemented the possibility to suggest resources to be preloaded and prefetched in priority via a simple Hypertext Markup Language (HTML) attribute. HTTP2 also added the possibility to the server to push resources directly to the client without the need to request them.

In the field of web mapping, these methods are hard to apply alone without wasting bandwidth. Indeed the map navigation in three dimensions would mean multiple level of preloading or prefetching. In this area, Google published a library using Machine Learning to try to predict user behaviour in terms of navigation (Gechev, Osmani, Hempenius, & Mathews, n.d.). Currently the library try to anticipate which page the user will choose to display after the current one and prefetch it, but we can imagine that such approaches could be applied to web mapping in the future and avoid issues like in Figure 6.

Leverage the Server

Web mapping fields use since a long time ago the tilling technique which subdivide an image into a grid of multiples small square called tile and produce a pyramid of grids to serve these tiles according to the third dimension – the zoom. Since 1999, we use Web Map Service (WMS) servers to provide use with these grid of bitmaps. The method, still actively used, bring a relative simplicity to the web browser with the only job to rearrange the tiles in a grid without any calculations. Nowadays, it is essentially used to display base layers like terrestrial images or a background map. However some organisations present markers or clusters of markers on a map with it. The technology calculate the position of the markers and renders the markers into a transparent grid of bitmaps which can be added on top of base layers. Basic interaction with markers is achieved with a request on the server and the coordinates of the click on the map as parameters. Loading time and transmission issues are avoided, but this technique also has disadvantages. The client has a reduced flexibility. When there is a need to filter or customise markers, the only way is to request the server for new bitmaps. In the example of Figure 8, we can observe that each zoom requires to load new images which create an experience of Markers Flashing.

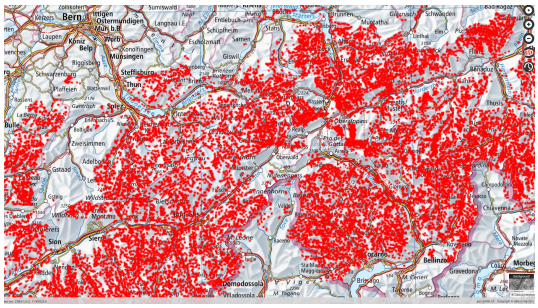


Figure 8. Video example of Markers Flashing with WMS
https://doi.org/10.6084/m9.figshare.9332309
https://map.geo.admin.ch (Terrestrial images swisstopo black and white), © Données:swisstopo

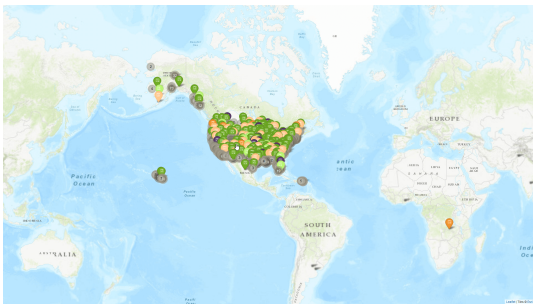


Figure 9. Video Case Study of Geocluster Drupal Plugin
https://doi.org/10.6084/m9.figshare.9331547
https://www.vistacampus.gov/map, Leaflet — Tiles © Esri

Through the years, experiments have been envisioned to leverage the server while still using the potential of the client (Ortelli, 2013; Dabernig, 2013; Adam, 2017; Rezaei & Fränti, 2018), though none was ever democratised. The usual gap between academic research and real application may explain the situation. Moreover some of this research were too specific or missed a good practical documentation.

These days, the popularity of the Node.js framework might change behaviour in the future since it employs JavaScript – the client-side programming language. For example, supercluster was designed to be client-side, but Agafonkin (2018) mentioned that it could be implemented on a server. In some situation, workflow would need to be rethought to link the library with the geospatial database instead of a single file.

In the Drupal community, the Geocluster plugin introduced by Dabernig (2013) has been presented into a case study at a Drupal technical conference (Paul, 2015). On the one hand, the demonstration shows a good performance in response time, but, on the other hand, as the interactivity still suffer with sluggish rendering (Fig. 9). The only responsible left to blame is the Document Object Model (DOM) on the client-side which takes care of the HTML markers. In another experiment, (Nebel, 2018) tried to generate this markup on the server rather than on the client-side, but he terminated in no time seeing that the performance was even poorer.

Rendering

In the previous sections, we noticed that the majority of listed user experience issues in chapter "Perceived performance issues" can be solved excepting Markers Flashing (Fig. 3) and Rendering Latency (Fig. 4) issues. In this section, we will cover the last level of optimisation which happens exclusively on the client-side.

Enhancing the DOM

In the last decade, one major attempt was made to improve the DOM. The concept of virtual DOM appeared several years ago, but was mainly popularised by React, the JavaScript framework which uses it as a primary argument to promote their technology. The principle is to create an abstraction of the DOM in memory and synchronise only needed changes. The whole concept was misunderstood and advertised by third parties to something which would be faster than the DOM itself. As a matter of fact, virtual DOM allows developers to build applications with a good default performance, but it does not improve in any way the performance of the DOM (Harris, 2018). This outcome is presumably also relevant to all alternatives of virtual DOM like Memoized DOM, hyperHTML, etc.

The DOM has a serious reputation to be slow among developers, but benchmark demonstrations like Teller (2018) explains that competences and knowledge of its functioning are key for good performance while also confirming the benefit of virtual DOM to reduce the risk of extremely poor performance. However, the notion of DOM is relatively old and was not built for applications like appending thousands of points on a page. Recommendations issued by Google for Web developers argue for a maximum 1,500 nodes per page ("Uses An Excessive DOM Size — Tools for Web Developers", 2019). Naturally, clustering methods help to mitigate the problem by limiting the numbers of nodes, but circumstances where the number of nodes rapidly inflates are not extraordinary and perceived performance issues appear (e.g. Figure 9 has 2,071 nodes at the time of writing, visually way more than at the time of Paul, 2015 publishing). Therefore alternatives should be taken into consideration.

Drawing Markers

The revolution and alternative to the DOM came as early as 2011 with the WebGL API. Not to be confused with the Canvas API, the technology allows using the Graphics Processing Unit (GPU) within the web, whereas the Canvas API is older (2004), limited to two dimensions and not hardware-accelerated. A frequent confusion comes from the fact that both are rendering graphics inside the HTML canvas element. Today, the Canvas API tends to be left behind in favor of WebGL or to be used for compatibility reasons. The adoption of WebGL inside the web community was mainly driven by the conception of intermediary JavaScript libraries like three.js or Pixi.js which tries to translate the complex language behind WebGL. Talking to the GPU require to use a special language – the OpenGL Shading Language (GLSL) – and to tessellate data into triangles. In practice, the adoption of WebGL in web cartography is still in its infancy, but the Mapbox company is well determined to change this. The use of WebGL in web cartography is highly linked to the research for a vector tiles specification. One of the main reasons given to this innovation, saw as a replacement of WMS, is the possibility to style elements directly in the browser. Aside from this main characteristic and in terms of user experience, the use of WebGL allow dynamic rendering mentioned by Eriksson and Rydkvist (2015). Regular mechanisms used to render HTML markers try to keep a seamless experience between level of zooming by using some transitions on current zoom data until the next zoom data have been charged in memory. Vector data eliminate this kind of

workaround with their ability to be rendered multiple time according to the zoom without any additional network requests. Markers Flashing (or flickering) and Rendering Latency are overcome with this only ability (Fig. 10). The experience can even more easily enhance with for example Progressive Fading (Fig. 11). Moreover, WebGL bring an evident advantage to our research of an optimised experience. All points rendered with WebGL are displayed in a unique HTML element which renders like a simple bitmap, allows theoretically unlimited of points (by using instancing property of WebGL2 - Ninomiya, 2017) and therefore remove the risk of too many markers performance issues.

Meanwhile version 2 of WebGL is only supported by a limited number of web browsers, multiple proposals have been initiated for the future replacement of WebGL. The efficiency and reliability of WebGL is contested from all parts and there is multiple need like accessing low-based GPU capacity, multithreading or access to the GPU memory (Weissflog, 2016). Apple, Google and Mozilla are developing and experimenting their own proposal for an API, all encompass with the project working name of WebGPU. Few years will be necessary first to see a generic implementation and above all to measure the impact on web cartography.

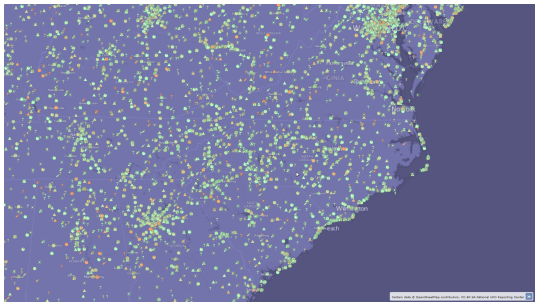


Figure 10. Video example of OpenLayers Icon Sprites with WebGL
https://doi.org/10.6084/m9.figshare.9332366
https://openlayers.org/en/master/examples/icon-sprite-webgl.html, Certain data © OpenStreetMap contributors, CC-BY-SA
National UFO Reporting Center

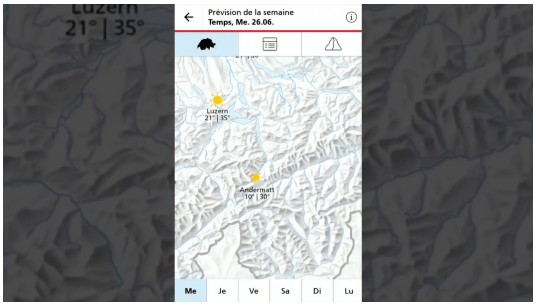


Figure 11. Video example of Progressive Markers Fading
https://doi.org/10.6084/m9.figshare.9331865
MeteoSwiss version 2.4.1 on Android 8.0.0

DISCUSSION

With WebGL, the last piece of the puzzle, we saw that the opportunity to build and display a web map with a multitude of markers are boundless. Most limitations on user experience have been cleared away. In this last section, we will explore two questions: why technologies discussed in this paper are not widely adopted and why we should rethink the usage of aggregation in projects involving web mapping.

Towards More Documented Workflows

For a long time, web mapping was restrained to the server with technology such as WMS. However year over year, JavaScript which has been for a long time mocked became a compelling language and opened an era of pure client-side calculations and rendering. Today, a small movement is going towards the perspective of efficiently combine and reunite server and client. Prominent JavaScript frameworks like Vue.js are encouraging the adoption of Server Side Rendering (SSR) methods which give an extra channel of communication between the two parts of the model.

In web cartography, technologies are available and accessible, but plenty of work has to be done in order to have a consistent workflow or at least document and explain how such a workflow should be achieved. Even though the first implementation in web browsers of WebGL is almost ten years old, the usage of this particular technology in web mapping is relatively narrow. The causes could be found on the side of the Leaflet's usage — the biggest and the most popular web cartography library (according to the stars ranking on Github.com). The reason behind its success lie in the relative simplicity of the implementation and functioning. In the presentation of Agafonkin (2015), the creator of Leaflet explains that in order to keep this characteristic, WebGL will not be implemented at the heart of the library. From

another perspective, we cannot disregard the fact that Agafonkin is an employee of Mapbox and a major contributor of Mapbox GL. Economic reasons are at stake and it seems reasonable to privilege the product of their company over alternative and independent libraries. Today, the situation is identical except that some third-party plugin has been developed like PixiOverlay which adds a layer of WebGL on Leaflet. However, user experience is less refined than Mapbox GL and still suffers from the example of Triggering Delay (Fig. 13). Beyond the addition of a generic WebGL overlay, no straightforward solution for Leaflet has been published to bring a solution which would render clusters and not only markers. Concurrently, Mapbox has successfully implemented a workflow in their Mapbox GL library using their supercluster library client-side and a process to convert the resulting clusters and points into their own Mapbox vector tiles specification (Fig. 12). Furthermore, Mapbox generously created multiple open source tools (like GeoJSON-vt, vt-pbf, tippecanoe, etc.) for their stack solutions which provide a complete end-to-end for serving clusters of markers efficiently by taking advantage of both client and server. Therefore, it becomes clear that the only missing part would be a proper practical documentation allowing anybody to reconstruct a similar stack in the easiest way possible and change with time the general behaviour of developers who are not specialist in this field or rarely implement cartography.

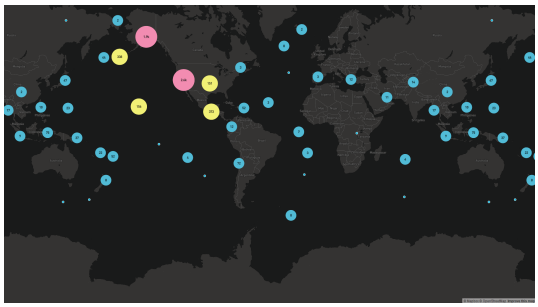


Figure 12. Video of Mapbox GL Clustering
https://doi.org/10.6084/m9.figshare.9332795
https://docs.mapbox.com/mapbox-gl-js/example/cluster/, © Mapbox © OpenStreetMap

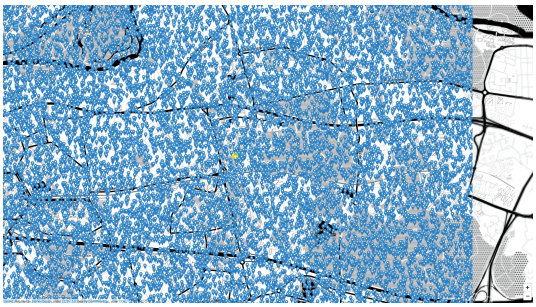


Figure 13. Video of Leaflet.PixiOverlay – One million markers Demo
https://doi.org/10.6084/m9.figshare.9332153
https://manubb.github.io/Leaflet.PixiOverlay/many-markers.html, Leaflet — Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

Questioning the Systematic of Clustering

The overview of technologies and potential optimisations with especially the use of WebGL put into questions the reason of performance for choosing aggregation, outlined in the second chapter of this paper. Clustering aggregation tends to be systematically applied in a situation where POIs need to be displayed on a map without really judging the necessity. Figure 14 illustrates this tendency well. In this example, clusters are everywhere at low levels of zoom, albeit when we zoom in, icons of clusters and individual markers are mixing and visually obstruct the field of view. The user cannot anymore quickly compare the areas and find the most active region in the capture of mosquitoes, for instance.

As stated by Nemeth (2015), the meaning of clustering all too often lack of sense for many reasons, but mainly because grouping points together with variable distance misrepresent density as long as the cluster position is only the result of mathematical depiction. Additionally clustering can become a source of inconvenience. In some situation, it introduces other problems like the obfuscation of the distance between points illustrated by Figure 5. In practice, developers or designers tend to make minimal customisation of the cluster icon resulting in a lack of proper comparison attributes like variable sizes or colours other than just numbers in a text label (Fig. 5). Some system like Google MarkerClusterer encourage the use of different colours and restrict the possibility to change the size of the cluster icon in percentage to the number of points that it contains (Fig. 14). A majority of maps represent POIs with the Google Maps Pin or an imitation despite the research of other ways to transmit the meaning like the concept of Generative Marker (Meier, 2016) which exhort to use representative markers of the data and to display different icons to reflect the number of elements behind a cluster.

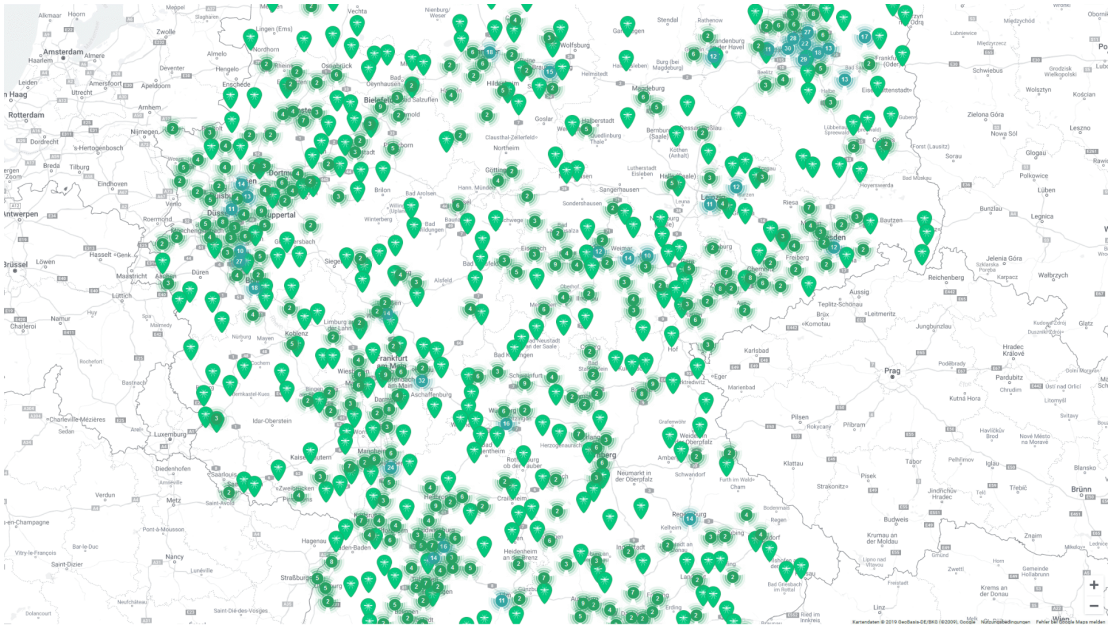
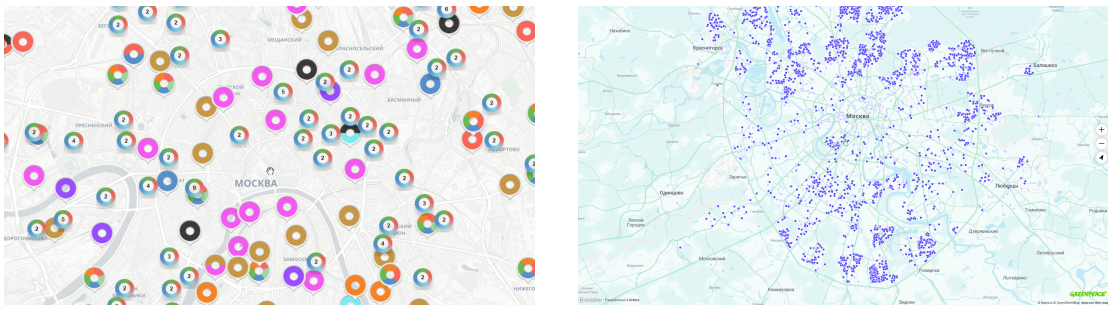


Figure 14. The Mosquito Atlas of Germany 2018
https://mueckenatlas.com/karte-der-sammler-2018, Kartendaten © 2019 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional

Nevertheless, clustering in some exceptional case can be meaningful. When this aggregation method is used to depict a sum of attributes inside a common depicted area, for example. Region-aware clustering method exist as an alternative to mathematical one. In this case, aggregations are constructed with a database of territory like countries, states or smaller (administrative) area.

After all in terms of clutter reduction methods, aggregation is only one of the height methods according to the classification of Korpi and Ahonen-Rainio (2013). Some libraries are exploring other clutter reduction methods like animation of the overlapping markers ("Ordering Overlapping Markers - Maps API for JavaScript", n.d.). However recently in the usage and with the power of WebGL, alternatives have been chosen over clustering (Fig. 15). The trend of 'symbolisation simplified' method will certainly become the norm. A lot of examples would benefit from it, but the change of development habits will take a while longer.



(a) Current map with clustering method
https://doi.org/10.6084/m9.figshare.9331985
https://recyclemap.ru, Map data © 2019 Google

(b) Beta map with symbolisation simplified method
https://doi.org/10.6084/m9.figshare.9332051
https://beta.recyclemap.ru, © Mapbox © OpenStreetMap

Figure 15. The transition of clutter reduction method on the Greenpeace Russian recycling map project

A Note on Accessibility

Eventually, we cannot finish this technical overview without making a general reminder of accessibility which is often left behind whether it is by designers or developers. Every web professional should be aware of two aspects before diving into practical work.

First, the choice of technology may have an impact on accessibility. Although WebGL has interactive functionalities like click recognition and mapping, it would be an illusion to think that it has the same accessibility level as the DOM. Basically, the construction of the DOM allows the user to access each node with the keyboard whereas WebGL, as mentioned before, build a static bitmap. Hit region functionalities (Sadecki, 2014; "Hit regions and accessibility", n.d.) which would give the ability to navigate with the keyboard is only supported on two major browsers currently. The proposed and promising Accessibility Object Model (AOM) will certainly make a great difference in this situation. This JavaScript API will allow developers to create virtual nodes exclusively for the accessibility purpose and leave the DOM as it is. However if keyboard navigation is an essential mechanism for accessibility in a standard web page, an abundance of markers or clusters are not easy to review without hierarchies. Mapbox is currently experimenting with the development of a plugin which converts important markers to HTML nodes ("mapbox-gl-accessibility - An accessibility control for Mapbox GL JS.", 2019), but defining the importance of markers is largely dependent on a use case and it is difficult to generalise.

Google has implemented another form of accessibility combination their Google Maps Platform. The map view is sliced in the region or area like a city and the user is invited to select an area. Blind users also benefit of this mechanism, because each area is named and have a common sense. Though, this kind of technique is difficult to replicate and, like the experimentation of Mapbox, might not suit every usage. In the end, it should be noted that keyboard navigation on a map is experimental or very tailored to a use.

Secondly, we went through a lot of web map during the writing of this paper, but only one (Fig. 1 and Fig. 2) differentiate cluster icons with colours as well as with their shape. People with low-vision or colour blindness will gain a lot if it were expanded.

CONCLUSION

A large spectrum of approaches to enhance perceived performance of clustering in web mapping was proposed and discussed in this paper. Most of them are available for several years, but there is a certain reluctant to use it. Furthermore, the best performance is a matter of combining technologies. We think that straightforward documented workflows are needed to change this general behaviour.

Above all, whether the performance was one of the main reasons to adopt clustering in web mapping, we exposed through examples that the introduction of new techniques and technology make this statement obsolete. No clutter reduction method is perfect, though web practitioners need to rethink their usage of clustering and enlarge their vision of the possibilities to reduce cluttering.

ACKNOWLEDGMENTS

The present thesis was written as a part of the Certificate of Advanced Studies in Interaction Science and Technology (http://human-ist.unifr.ch/cas) organised by the Human-IST (Human-Centered Interaction Science and Technology) Institute of the University of Fribourg. Thank you to the Media Engineering Institute (MEI), the School of Management and Engineering Vaud (HEIG-VD) and the Human-IST (Human-Centered Interaction Science and Technology) Institute of the University of Fribourg who would have made possible this writing.

REFERENCES

References can be found in the following Zotero Public Group. Everyone is welcome to contribute or use the group for others works: https://www.zotero.org/groups/2327704

Adam, P. (2017, September 3). Pre-clustering google maps markers using KMeans in django [Medium]. Retrieved June 20, 2019, from https://medium.com/@padam0/pre-clustering-google-maps-markers-using-kmeans-in-django-aeabc6eb2c0b

Agafonkin, V. (2015, March 19). Leaflet, WebGL and the future of web mapping. FOSS4G North America 2015. Retrieved May 29, 2019, from https://www.youtube.com/watch?v=lj4SS1NTqyY

Agafonkin, V. (2016, March 31). Clustering millions of points on a map with supercluster [Points of interest - the official mapbox blog]. Retrieved May 24, 2019, from https://blog.mapbox.com/clustering-millions-of-points-on-a-map-with-supercluster-272046ec5c97

Agafonkin, V. (2017, April 27). A dive into spatial search algorithms [Points of interest - the official mapbox blog]. Retrieved July 8, 2019, from https://blog.mapbox.com/a-dive-into-spatial-search-algorithms-ebd0c5e39d2a

Agafonkin, V. (2018, July 3). Client-side clustring or server-side clustring · issue #91 · mapbox/supercluster [GitHub]. Retrieved July 10, 2019, from https://github.com/mapbox/supercluster/issues/91#issuecomment-402030878

Alstad, K. (2015, April 15). New findings: State of the union for ecommerce page speed and web performance [spring 2015] [Radware blog]. Retrieved July 1, 2019, from https://blog.radware.com/applicationdelivery/wpo/2015/04/new-findings-state-union-ecommerce-page-speed-web-performance-spring-2015/

Bernstein, M. (2014, June 5). 5 reasons to use protocol buffers instead of JSON for your next service [Codeclimate]. Retrieved July 5, 2019, from https://codeclimate.com/blog/choose-protocol-buffers/

Burigat, S. & Chittaro, L. (2008). Decluttering of icons based on aggregation in mobile maps. In *Map-based mobile services* (pp. 13–32). Springer. doi:10.1007/978-3-540-37110-6_2

Chakraborty, S., Nagwani, N. K., & Dey, L. (2014, June 18). Performance comparison of incremental k-means and incremental DBSCAN algorithms. *arXiv:1406.4751 [cs]*. arXiv: 1406.4751. Retrieved July 2, 2019, from http://arxiv.org/abs/1406.4751

Cluster analysis. (2019, June 16), In *Wikipedia*. Page Version ID: 902151874. Retrieved July 8, 2019, from https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=902151874

Dabernig, J. (2013, June 2). *Geocluster: Server-sideclustering for mapping in drupal based on geohash* (Master Thesis, Fakultät für Informatik der Technischen Universität Wien, Vienna). Retrieved May 10, 2019, from http://dasjo.at/files/geocluster-thesis-dabernig.pdf

DBSCAN. (2019, April 18), In *Wikipedia*. Page Version ID: 893049545. Retrieved May 17, 2019, from https://en.wikipedia.org/w/index.php?title=DBSCAN&oldid=893049545

Delort, J. (2010, February). Vizualizing large spatial datasets in interactive maps. In *2010 second international conference on advanced geographic information systems, applications, and services* (pp. 33–38). 2010 second international conference on advanced geographic information systems, applications, and services. doi:10.1109/GEOProcessing.2010.13

Door, S. (2012, August 17). Instagram komt met versie 3.0 [GSMacties.nl]. Retrieved July 2, 2019, from https://www.gsmacties.nl/nieuws/2012/08/instagram-komt-met-versie-3-0/

Eriksson, O. & Rydkvist, E. (2015, August 26). *An in-depth analysis of dynamically rendered vector-based maps with WebGL using mapbox GL JS* (Master Thesis, Linköping University). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-121073

Gechev, M., Osmani, A., Hempenius, K., & Mathews, K. (n.d.). Guess.js. Retrieved July 9, 2019, from https://guess-js.github.io/docs

Gibbs, S. (2015, February 8). Google maps: A decade of transforming the mapping landscape. *The Guardian*. Retrieved June 27, 2019, from https://www.theguardian.com/technology/2015/feb/08/google-maps-10-anniversary-iphone-android-street-view

Harris, R. (2018, December 27). Virtual DOM is pure overhead [Svelte.js - blog]. Retrieved May 29, 2019, from https://svelte.dev/blog/virtual-dom-is-pure-overhead

Hit regions and accessibility. (n.d.). Retrieved July 13, 2019, from https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Hit_regions_and_accessibility

Huang, H. & Gartner, G. (2012). A technical survey on decluttering of icons in online map-based mashups. In *Online maps with APIs and WebServices* (pp. 157–175). Springer. doi:10.1007/978-3-642-27485-5_11

Korpi, J. & Ahonen-Rainio, P. (2013, August 1). Clutter reduction methods for point symbols in map mashups. *The Cartographic Journal*, *50*(3), 257–265. doi:10.1179/1743277413Y.0000000065

Krebs, B. (2017, January 31). Beating JSON performance with protobuf [Auth0 blog]. Retrieved July 5, 2019, from https://auth0.com/blog/beating-json-performance-with-protobuf/

Larsgård, N. (2017, April 9). Comparing sizes of protobuf vs json [Random code]. Retrieved July 5, 2019, from https://nilsmagnus.github.io/post/proto-json-sizes/

Lee, J.-G. & Kang, M. (2015, June 1). Geospatial big data: Challenges and opportunities. *Big Data Research*. Visions on Big Data, *2*(2), 74–81. doi:10.1016/j.bdr.2015.01.003

Mahe, L. & Broadfoot, C. (2010, December). Too many markers! - google maps API — google developers. Retrieved July 1, 2019, from https://web.archive.org/web/20121113185947/https://developers.google.com/maps/articles/toomanymarkers

mapbox-gl-accessibility - An accessibility control for Mapbox GL JS. (2019, May 9). Retrieved May 29, 2019, from https://github.com/mapbox/mapbox-gl-accessibility

McConnell, J. (2017, November 13). WebAssembly support now shipping in all major browsers [The mozilla blog]. Retrieved July 8, 2019, from https://blog.mozilla.org/blog/2017/11/13/webassembly-in-browsers

Meert, W., Tronçon, R., & Janssens, G. (2006). *Clustering maps* (Master Thesis, Katholieke Universiteit Leuven, Leuven). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.132.6977&rep=rep1&type=pdf

Meier, S. (2016). The marker cluster: A critical analysis and a new approach to a common web-based cartographic interface pattern. *International Journal of Agricultural and Environmental Information Systems (IJAEIS)*, *7*(1), 28–43. doi:10.4018/IJAEIS.2016010102

Nebel, P. (2018, January 30). Dynamic server-side clustering for large datasets [Geovation tech blog]. Retrieved July 10, 2019, from https://geovation.github.io/dynamic-server-side-geo-clustering

Nemeth, A. (2015, March 21). ”how do i do clustering on a map correctly” is a common question in mapping applications. [User experience stack exchange]. Retrieved July 2, 2019, from https://ux.stackexchange.com/a/75190/116603

Newton, C. (2016, September 6). Instagram is getting rid of photo maps [The verge]. Retrieved July 2, 2019, from https://www.theverge.com/2016/9/6/12817340/instagram-photo-map-removals

Ninomiya, K. (2017, May 17). Webinar recap: WebGL 2.0, what you need to know [The khronos group]. Retrieved July 12, 2019, from https://www.khronos.org/blog/webinar-recap-webgl-2.0-what-you-need-to-know

Ordering Overlapping Markers - Maps API for JavaScript. (n.d.). Retrieved July 13, 2019, from https://developer.here.com/api-explorer/maps-js/markers/ordering-overlapping-markers

Ortelli, G. (2013, August 1). Server-side clustering of geo-points on a map using elasticsearch [Trifork blog]. Retrieved May 22, 2019, from https://blog.trifork.com/2013/08/01/server-side-clustering-of-geo-points-on-a-map-using-elasticsearch/

Osmani, A. (2018, July 25). Speed is now a landing page factor for google search and ads! https://bit.ly/search-ads-speed … ”both efforts are leveraging real-world user experience data to prioritize and highlight web pages that deliver fast user experiences” evaluate perf with lighthouse & PageSpeed insights pic.twitter.com/6mzhwdrfkn [@addyosmani]. Retrieved July 1, 2019, from https://twitter.com/addyosmani/status/1022005088058073088

Paul, E. (2015, June). Case study: Large-scale, server side mapping with the leaflet-geocluster stack — twin cities drupal camp 2015. University of Minnesota Law School. Retrieved July 10, 2019, from http://2015.tcdrupal.org/session/case-study-large-scale-server-side-mapping-leaflet-geocluster-stack

Poskanzer, J. (2005). Clusterer.js - marker clustering routines for google maps apps. Retrieved June 27, 2019, from http://www.acme.com/javascript/Clusterer2.js

Rasmussen Eilstrup, J. (2014). Jens Eilstrup Rasmussen. Google Maps Pin. 2005 — MoMA [The Museum of Modern Art]. Retrieved June 28, 2019, from https://www.moma.org/collection/works/174200

Rezaei, M. & Fränti, P. (2018, November 30). Real-time clustering of large geo-referenced data for visualizing on map. *Advances in Electrical and Computer Engineering*, *18*, 63–74. doi:10.4316/AECE.2018.04008

Sadecki, M. (2014, May 15). *Canvas accessibility - past, present and future*. Global Accessibility Awareness Day (GAAD. Retrieved June 20, 2019, from https://www.w3.org/Talks/2014/0510-canvas-a11y/#1

Sandvik, B. (2012, November 18). How to minify GeoJSON files? [Mastermaps]. Retrieved July 4, 2019, from http://blog.mastermaps.com/2012/11/how-to-minify-geojson-files.html

Schoors, L. & Deveria, A. (n.d.). Can i use... HTTP/2 protocol [Can i use... support tables for HTML5, CSS3, etc]. Retrieved July 5, 2019, from https://caniuse.com/#search=http2

Spatial database: Spatial index. (2019, May 1), In *Wikipedia*. Page Version ID: 895018107. Retrieved May 24, 2019, from https://en.wikipedia.org/wiki/Spatial_database#Spatial_index

Svennerberg, G. (2009, May 6). Handling large amounts of markers in google maps – in usability we trust [In usability we trust]. Retrieved June 28, 2019, from http://www.svennerberg.com/2009/01/handling-large-amounts-of-markers-in-google-maps/

Teller, S. (2018, February 27). Building an interactive DOM benchmark, preliminary results [A geek with a hat]. Retrieved July 11, 2019, from https://swizec.com/blog/building-interactive-dom-benchmark-preliminary-results/swizec/8219

Tihonov, I. (2014, November 12). Speed up web maps - minify geojson. Retrieved May 29, 2019, from http://igortihonov.com/2014/11/12/speedup-web-maps-minify-geojson/

Urbica. (2018, July 6). Visualising large spatiotemporal data in web applications [Medium]. Retrieved June 20, 2019, from https://medium.com/@Urbica.co/visualising-large-spatiotemporal-data-in-web-applications-8583cf21907

Uses An Excessive DOM Size — Tools for Web Developers. (2019, May 29). Retrieved July 11, 2019, from https://developers.google.com/web/tools/lighthouse/audits/dom-size

Wagner, J. (2017, August 16). Using the paint timing API [CSS-tricks]. Retrieved July 1, 2019, from https://css-tricks.com/paint-timing-api/

Wagner, J. & Osmani, A. (2019, May 29). Reduce JavaScript payloads with code splitting — web fundamentals [Google developers]. Retrieved July 10, 2019, from https://developers.google.com/web/fundamentals/performance/optimizing-javascript/code-splitting/

Weissflog, A. (2016, August 13). Thoughts about a WebGL-next [The brain dump]. Retrieved July 12, 2019, from https://floooh.github.io/2016/08/13/webgl-next.html

Wen, T. (2017, April 21). Is protobuf 5x faster than JSON? [Dzone performance zone]. Retrieved July 5, 2019, from https://dzone.com/articles/is-protobuf-5x-faster-than-json-part-ii