International Cartographic Association
Association Cartographique Internationale

Comparative performance analysis of open source web mapping libraries for rendering large amounts of vector data

Dániel Balla [a,*], Mátyás Gede [a]

[a] *Institute of Cartography and Geoinformatics, ELTE Eötvös Loránd University, Budapest, Hungary, balla.daniel@inf.elte.hu, gedematyas@inf.elte.hu*

* Corresponding author

Keywords: performance analysis, vector, web maps, web cartography, open source, Leaflet, OpenLayers, Mapbox GL JS, MapLibre GL JS

Abstract:

These days, various web mapping technologies are available for creating interactive web maps. Although using these libraries is pretty easy and straightforward, creating cartographically comprehensive, clean and visually appealing maps still requires both cartographic expertise and programming skills. An important aspect of creating maps with these libraries is rendering performance. Today, web maps are consumed on devices capable of differing computational power and performance, therefore choosing the best-performing library for a particular dataset gets greater emphasis. Previous literature of web mapping library performance analyses mostly approached the subject tailored to a specific use case or visualization technique. (Netek et al., 2019) conducted a comparative study on the libraries based on clustering and heatmap visualization performance, (Ježek et al., 2016) compared the performance of a self-developed GPU-based approach for heatmap visualization to other solutions provided by plugins, while (Zunino et al., 2020) evaluated the execution time and network usage of Leaflet, OpenLayers and Mapbox GL JS for raster and vector data simultaneously, using a Life Quality Index (LQI) web map on 14 different mobile devices. In contrast, this paper focuses on native performance of rendering vector data, based on datasets containing gradually increasing number of features of the same type. Furthermore, these papers omitted MapLibre GL JS in the analyses, while it is included here.

The aim of this research is to assess and compare the loading and rendering performance of readily available, open source web mapping libraries, specifically for vector data. We aim to provide web map developers with quantitative performance information for rendering different amounts of features. Four popular JavaScript libraries have been evaluated in terms of rendering performance for large amounts of vector data: Leaflet, OpenLayers, Mapbox GL JS and MapLibre GL JS. The latest version of libraries available at the end of 2024 were tested. Precisely assessing their performance by measuring loading and rendering times is convoluted, as the libraries all do various internal processes asynchronously and determining render completion is not evident. They do not offer events related to render completion that get fired in a consistent manner. To accommodate this and measure the temporal range between reading the dataset file and the features appearing on the map, the Chrome DevTools Performance panel was carefully examined after each run to delineate library processes of interest. With numerous constraints taken into consideration to regulate and keep the analysis and comparison as consistent between the libraries as possible, the tests were conducted in a local environment on a desktop computer equipped a dedicated GPU, a local webserver and Google Chrome 131 as the browser. The workflow involved time measurements to a precision of a millisecond, the results were averaged over 10 runs for each dataset sample, resulting in 960 measured runs. To ensure a coherent analysis, a range of GeoJSON datasets were created for each feature type, only differing in the number of features within a type, while also minimizing other differences like vertex count or feature inconsistency among the datasets. Nine dataset variants, based on feature count were created, with 50, 100, 500, 1000, 5000, 10000, 50000, 100000 (points and lines only) and 500000 (points only) features. This way, 24 sample datasets were created.

Preliminary results show that generally, regardless of feature types, Leaflet and OpenLayers both excel at small number of features (up to 10k). Between them, Leaflet has an advantage over OpenLayers and is approximately 1.15x faster for feature amounts up to the 1000 mark. Mapbox GL JS is generally faster than MapLibre GL JS in all tested scenarios by 70-200 ms for up to 5000 features, where the difference is down to approx. 60 ms between them. The results of testing specifically with point features using SVG symbols indicate that, while for points up to 5000 Leaflet and OpenLayers still excel (140 and 115.5 ms rendering time, respectively), for 10k points the four libraries' performance somewhat converge. For 50k and more points, Mapbox GL JS renders them the quickest, with OpenLayers not far behind (466.9 and 588.4 ms, SD = 15.96 and 8.45, respectively). MapLibre GL JS is generally slower throughout the tests, and from

50k points onwards it joins Leaflet as the slowest libraries for lots of points, taking roughly twice the time as the other two. Rendering performance of lines and polygons were similar, but different to those of points. Up to 50k lines and 10k polygons, Leaflet and OpenLayers were the fastest in all tested scenarios. OpenLayers was substantially fast with 100k lines, almost twice as fast as all other libraries in that test, taking only 1295.5 ms (SD = 9.35) on average. For the largest sample with 50k polygons, Mapbox GL JS excelled with 1265 ms (SD = 9.35), while OpenLayers took 1366.4 ms (SD = 28.13) and Leaflet took 1539.9 ms (SD = 25.68). MapLibre GL JS was significantly slower than all other libraries for both lines and polygons, in all scenarios. One would expect that Mapbox GL JS and MapLibre GL JS both take advantage of GPU-acceleration of WebGL, but during these tests, their potential edge did not clearly come out.

A notable anomaly is, that while the performance of Leaflet and OpenLayers scale nicely with the increasing amount of features, for Mapbox GL JS and MapLibre GL JS this can not be stated. Results showed that for Mapbox GL JS and MapLibre GL JS, any effect of larger number of features on their performance is offset, only applicable to datasets with 1000 features and beyond. Under this amount, they render any number of features with a consistent speed with respect to themselves, the differences between runs on these datasets are well within margin of error (SD = 3-5). Regardless of feature count and layers, they initially load their map object elements in 160 and 350 ms, respectively, which compared to the other two are 130 and 315 ms slower. Although Mapbox GL JS relies on token authentication, this does not justify the slow map element initialization, since it is done asynchronously, and the request usually only takes a few ms. Moreover, MapLibre GL JS does not do any authentication and is even slower than Mapbox GL JS on initialization. This makes Mapbox GL JS and MapLibre GL JS much less suitable for rapid rendering of small number of features.

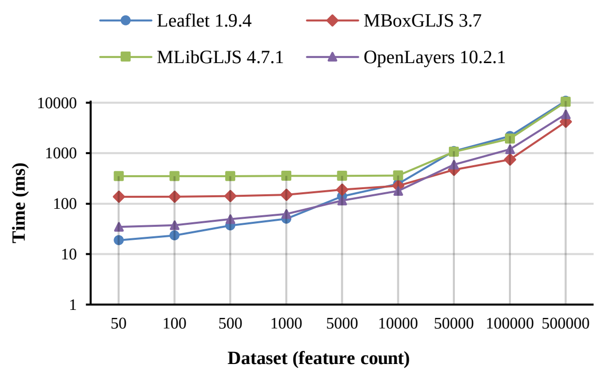

Figure 1. Results of testing with point features.

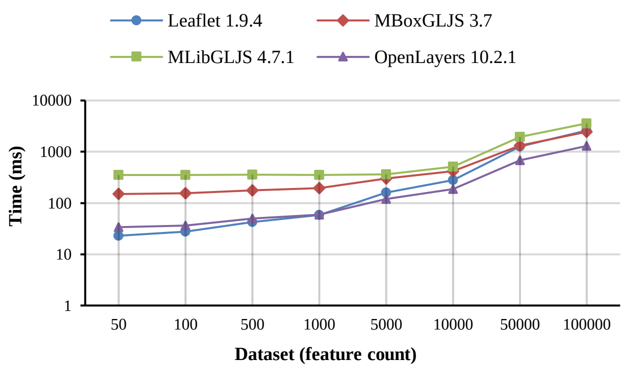

Figure 2. Results of testing with line features.

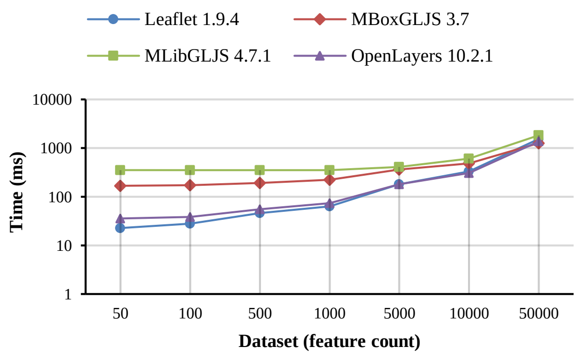

Figure 3. Results of testing with polygon features.

References

Ježek, J., Jedlička, K., Mildorf, T., Kellar, J., & Beran, D. (2016). Design and Evaluation of WebGL-Based Heat Map Visualization for Big Point Data. In: *The Rise of Big Spatial Data*, pp. 13–26. doi:10.1007/978-3-319-45123-7_2

Netek, R., Brus, J., & Tomecka, O. (2019). Performance Testing on Marker Clustering and Heatmap Visualization Techniques: A Comparative Study on JavaScript Mapping Libraries. In: *ISPRS International Journal of Geo-Information, 8*(8), p. 348. doi:10.3390/ijgi8080348

Zunino, A., Velázquez, G., Celemín, J., Mateos, C., Hirsch, M., & Rodriguez, J. (2020). Evaluating the Performance of Three Popular Web Mapping Libraries: A Case Study Using Argentina's Life Quality Index. In: *ISPRS International Journal of Geo-Information*, 9(10), p. 563. doi:10.3390/ijgi9100563