WebGraphViz: A WebGL-Based Interactive Graph Visualization Tool for Retail Analytics

Luis Miguel Aguilar, Ragaad AlTarawneh, Shah Rukh Humayoun†

Department of Computer Science, San Francisco State University, USA

Abstract

The growing volume and complexity of data from sources like social media, IoT systems, and security devices highlight the need for scalable, high-performance visualization tools. Traditional web technologies such as SVG and Canvas often struggle with large datasets, limiting interactivity. We present WebGraphViz, a WebGL-based graph visualization tool that leverages GPU parallelism to overcome these limitations. Performance evaluations across three interaction experiments in both high- and low-performance environments show that WebGraphViz significantly outperforms its SVG-based counterpart, enabling smooth exploration of large-scale graph data.

CCS Concepts

• ***Human-centered computing*** → ***Graph drawings;*** • ***Information systems*** → ***Graph-based database models;***

1. Introduction and Related Work

2. Introduction

The rapid growth of complex data from sources like social media, IoT systems, and security devices, often stored in the cloud, has heightened the need for effective visualization and query interfaces. As highlighted by [MAT23], extracting insights from large datasets poses major challenges. Interactivity is limited by query complexity, degrading user experience [DBCA24], while real-time scalability remains difficult [KAC21,MWO*20], as many platforms are optimized for smaller data volumes, resulting in high latency at scale.

Visual exploration of large-scale graph data faces persistent challenges, including high data volume, query latency, and limited interactivity [MAT23]. Prior studies have evaluated the performance of web technologies like SVG, Canvas, and WebGL for visualizing structured data [HKD18, Lin20, HK18], often finding WebGL more scalable. However, these evaluations were limited to small datasets (under 4,000 nodes), focused primarily on tree structures, and lacked diverse interaction tests or system configurations, highlighting the need for broader investigations involving complex graph structures, larger datasets, and varied hardware environments. Standard web technologies like SVG and HTML5 Canvas support complex and dynamic visualizations, but their rendering performance degrades with larger datasets, limiting scalability [Lin20]. This limitation is evident in GraDVis, an interactive graph visualization tool for Visual Data Management System (VDMS) [ZHATH24,GCRS17], which, despite enabling graph exploration, struggled with performance and interactivity on large datasets.

To address these challenges, we developed WebGraphViz, a WebGL-based visual analytics tool for exploring graph data in VDMS. By leveraging GPU acceleration, it improves rendering performance and real-time interactivity over previous SVG- and Canvas-based tools. The tool was evaluated via frames per second (FPS) measurements across three computational environments.

3. WebGraphViz System Implementation

WebGraphViz is a web-based tool designed to render and explore graph data from VDMS. It features a RESTful API backend built with Node.js, TypeScript, and Express, providing two main endpoints: one for retrieving and enriching full graph data with layout attributes (e.g., coordinates, labels, icons), and another for filtering customer-specific data to construct a tree hierarchy representing individual journeys. The frontend is developed with SigmaJS [mO24] and Graphology [Pli23] for handling graph rendering and manipulation, and D3.js [Obs24] enabling a collapsible tree visualization. Together, these components provide a responsive, scalable interface for interactive graph exploration.

The graph layout, depicted in Fig. 1(a), is rendered using React Sigma components and graphology properties. A force-directed layout with the Barnes-Hut approximation is employed, reducing the time complexity of the standard ForceAtlas algorithm from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$ for clustered nodes. To improve readability during panning, each node is assigned predefined colors, icons, sizes, and labels, as illustrated in Fig. 1(b). Additionally, clicking

† humayoun@sfsu.edu

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.2312/evp.20251137

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

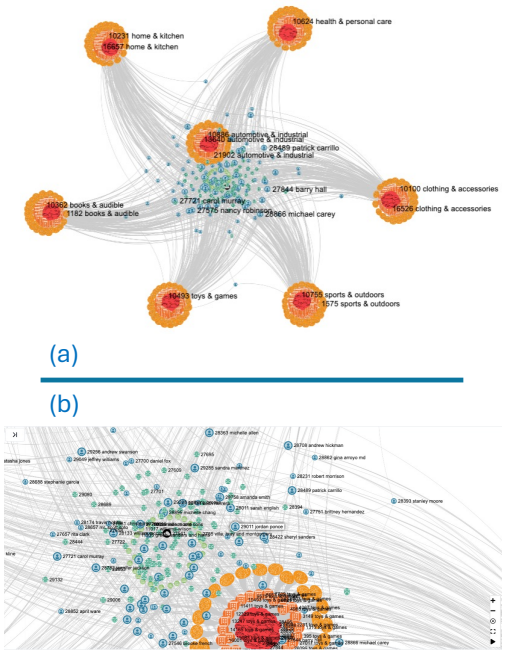

Figure 1: ***(a)*** *A force-directed graph comprising 29,000 nodes and 75,000 links, rendered using WebGL.* ***(b)*** *Nodes are grouped into clusters, distinguished by colors and icons representing node types. Node sizes are proportional to their degrees, and custom labels are applied to enhance readability.*

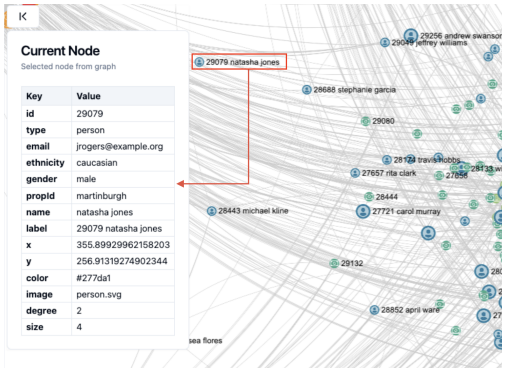

Figure 2: *Display of node properties upon user interaction with a node in the Fig. 1(b) graph.*

on a node reveals its properties, as shown in Fig. 2, enabling detailed analysis.

To evaluate our implementation's performance, we conducted three interaction experiments measuring average frames per second (FPS) across varying dataset sizes in two computational environments, one is SVG-based and the second one is WebGL-based: one SVG-based and the other WebGL-based. FPS was recorded using Chrome DevTools' "Frame Rendering Stats" in Microsoft Edge for its real-time performance metrics. We automated the tests using Macro Recorder to ensure consistency, executing predefined

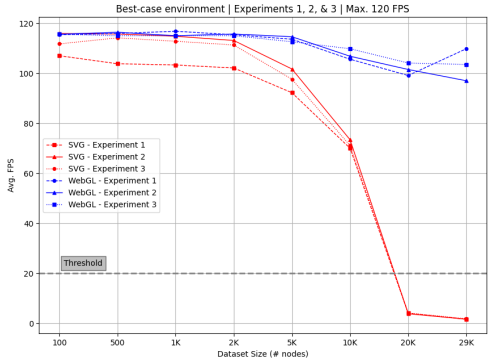

Figure 3: *In the best-case environment (120 FPS max), both implementations sustained over 100 FPS up to 5,000 nodes. As dataset size grew, SVG performance fell to near 0 FPS, while WebGL remained near 100 FPS.*

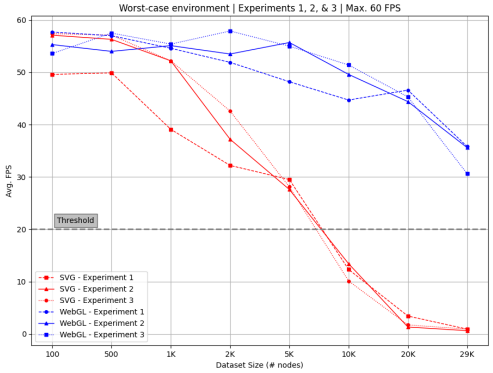

Figure 4: *In the worst-case environment (60 FPS cap), all three experiments showed both implementations achieving over 50 FPS up to 500–1,000 nodes. As size increased, performance declined: SVG dropped to near 0 FPS, while WebGL fell to around 30 FPS.*

macros for user interactions. The first experiment involved zooming to simulate typical exploration, the second combined zooming and focusing to reflect cluster-level navigation, and the third used more granular interactions targeting individual nodes for detailed comparison. We tested with set of graph datasets ranging from 100 nodes to 29,000 nodes, defining 20 FPS as the minimum threshold for acceptable interactivity [Lin20]. FPS was preferred over rendering time for its precision in capturing real-time responsiveness [HKD18].

Our results show that the WebGL implementation consistently outperformed the SVG-based version in average FPS across all experiments and environments. In both best- and worst-case scenarios (Fig. 3 and Fig. 4), FPS declined with increasing dataset size, but the SVG version dropped below the 20 FPS interactivity threshold, falling to near zero in the worst case, resulting in severe interface lag. In contrast, the WebGL version maintained FPS above the threshold, ensuring smooth interaction even with large datasets. In future work, we plan to extend our evaluation to larger graphs and more complex real-time visual queries.

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

References

[DBCA24] DHOLE K., BAJAJ S., CHANDRADEVAN R., AGICHTEIN E.: QueryExplorer: An interactive query generation assistant for search and exploration. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)* (Mexico City, Mexico, June 2024), Chang K.-W., Lee A., Rajani N., (Eds.), Association for Computational Linguistics, pp. 107–115. URL: https://aclanthology.org/2024.naacl-demo.11/, doi:10.18653/v1/2024.naacl-demo.11. 1

[GCRS17] GUPTA-CLEDAT V., REMIS L., STRONG C. R.: Addressing the dark side of vision research: Storage. In *9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 17)* (Santa Clara, CA, July 2017), USENIX Association. URL: https://www.usenix.org/conference/hotstorage17/program/presentation/gupta-cledat. 1

[HK18] HORAK M., KISTER M.: Comparing rendering performance of common web technologies for large visualizations. *Proceedings of the 2018 International Conference on Information Visualisation* (2018). 1

[HKD18] HORAK T., KISTER U., DACHSELT R.: Comparing rendering performance of common web technologies for large graphs. In *Poster Program of the 2018 IEEE VIS Conference* (2018). 1, 2

[KAC21] KUMAR S., ANDERSEN M. P., CULLER D. E.: Mr. plotter: Unifying data reduction techniques in storage and visualization systems, 2021. URL: https://arxiv.org/abs/2106.12505, arXiv:2106.12505. 1

[Lin20] LINDBERG A.: *Performance Evaluation of JavaScript Rendering Frameworks*. Master's thesis, Linköping University, Department of Computer and Information Science, 2020. 1, 2

[MAT23] MUNISWAMAIAH M., AGERWALA T., TAPPERT C. C.: Big data and data visualization challenges. In *2023 IEEE International Conference on Big Data (BigData)* (2023), pp. 6227–6229. doi:10.1109/BigData59044.2023.10386491. 1

[mO24] MÉDIALAB S.-P., OUESTWARE: Sigmajs, 2024. Accessed: October 18th, 2024. URL: https://www.sigmajs.org/. 1

[MWO*20] MA M., WU Y., OUYANG X., CHEN L., LI J., JING N.: Hivision: Rapid visualization of large-scale spatial vector data. *Comput. Geosci. 147* (2020), 104665. URL: https://api.semanticscholar.org/CorpusID:218973799. 1

[Obs24] OBSERVABLE: D3 by observable, 2024. Accessed: October 18th, 2024. URL: https://d3js.org/. 1

[Pli23] PLIQUE G.: Graphology, a robust and multipurpose graph object for javascript., Aug. 2023. URL: https://doi.org/10.5281/zenodo.8204237, doi:10.5281/zenodo.8204237. 1

[ZHATH24] ZAPATA J., HAIDER S. F., AL-TARAWNEH R., HUMAYOUN S. R.: Gradvis: A visualization tool for a visual data management system. In *Companion Proceedings of the 16th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (New York, NY, USA, 2024), EICS '24 Companion, Association for Computing Machinery, p. 89–91. URL: https://doi.org/10.1145/3660515.3662835, doi:10.1145/3660515.3662835. 1

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.