




WebGPU for Scalable Client-Side Aggregate Visualization

Gerald Kimmersdorfer¹ , Dominik Wolf¹ , Manuela Waldner¹ 

¹TU Wien, Institute of Visual Computing & Human-Centered Technology, Austria

Abstract

WebGPU is a new graphics API, which now provides compute shaders for general purpose GPU operations in web browsers. We demonstrate the potential of this new technology for scalable information visualization by showing how to filter and aggregate a spatio-temporal dataset with millions of temperature measurements for real-time interactive exploration of climate change.

CCS Concepts

• *Human-centered computing* → *Information visualization; Visualization systems and tools;*

1. Introduction and Related Work

With modern web technologies, information visualization has increasingly moved online as it is possible to create rich, engaging, and highly accessible interactive visualizations without having to install any software on the users' machines. JavaScript frameworks such as Data-Driven Documents (D3.js) [BOH11] and Vega-Lite [SMWH17] simplify the creation of interactive web-based visualizations. However, these popular frameworks are based on SVG, which scales poorly with the number of visualized data points compared to GPU-accelerated rendering APIs like WebGL [DPU*22].

Aggregate visualization [EF09] effectively limits the number of data points to be rendered by (hierarchically) summarizing data points into bins, where each bin shows an aggregate (e.g., average) of the enclosed data points. The number of elements to be rendered is thereby defined by the controllable bin resolution rather than the data size, and visual clutter can be effectively reduced. However, for interactive applications, recomputing the aggregates after filtering or zooming may become a bottleneck.

Filtering and aggregation can be effectively parallelized using general purpose GPU (GPGPU) methods, such as CUDA (e.g., [DPMO12]) or compute shaders (e.g., [SHCW22]). So far, however, GPGPU methods were not available for the web and therefore required a server and costly data transfer between render client and compute server for every user interaction. Pure client-side approaches thus required more sophisticated solutions for scalable aggregate visualization applications. For example, *im-Mens* [LJH13] and *P5* [LM19] use WebGL to store data as textures, which are then interpreted in parallel by a fragment shader. *Crossfilter* [cro15] uses sorted indices for rapid filtering and aggregation up to a few million data points. *Falcon* [MHH19] allows users to brush and link millions of data points within few seconds to milliseconds, using a resolution-sensitive indexing scheme for data tiles in combination with prefetching and progressive interaction.

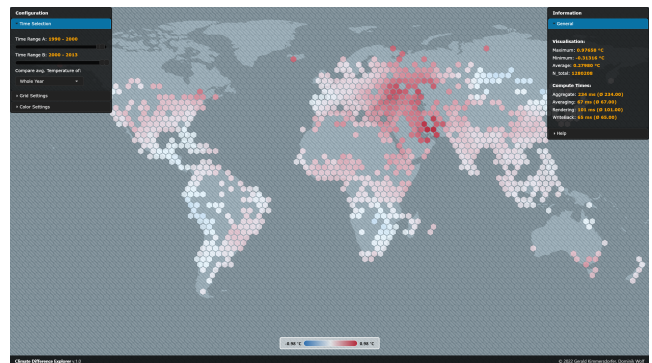


Figure 1: Our WebGPU-based Climate Change Explorer [cce23] performs filtering and hexagonal binning of millions of data points in real-time. The screenshot shows the average temperature difference between the years 1990-2000 and 2000-2013.

WebGPU is a new graphics and compute GPU API for the web. As of 2023, WebGPU is still under active development, with experimental support available in certain web browsers, while the specification continues to evolve. In addition to a streamlined shader pipeline that includes vertex and fragment shaders, WebGPU supports compute shaders as an enhancement over its predecessor, WebGL. With these new client-side capabilities, WebGPU has already attracted the attention by the visualization community. Usher and Pascucci [UP20] have found that WebGPU can achieve comparable performance as native Vulkan applications for large-scale scientific visualizations. Dyken et al. [DPU*22] presented the first WebGPU-based graph rendering framework, which uses compute shaders for parallel force-directed layout computations.

In this poster, we demonstrate the potential of WebGPU for aggregate visualization. We showcase the *Climate Change Ex-*

plorer [cce23] (Figure 1), a pure client-side spatio-temporal visualization application, which allows users to filter and aggregate millions of data points to explore temperature changes over selected time spans with interactive frame rates.

2. Climate Change Explorer

The *Climate Change Explorer* is an interactive web application that allows users to explore temperature differences between two specified time periods. The datasets used by the application are offered by the Berkeley Earth Organization [ber20]. The largest dataset is sourced by the *Local Station Data* and consists of 10.6 million monthly temperature averages for 5165 distinct locations (latitude and longitude).

We group the data into resizable hexagonal bins, with each bin corresponding to a distinct geographic region. Within each bin, data points are consolidated into a single value, which is then depicted by the hexagon’s color. This color represents the change in average temperature resulting from the comparison of the selected time ranges, with red indicating a higher average temperature in the second selected time range and blue indicating a lower temperature.

By default, we subdivide the world map into approximately 10,000 hexagonal bins, as shown in Figure 1. In our prototype, users have two options to interact with the visualization: 1) They can interactively change the time ranges to compare their average temperatures per bin and 2) they can modify the resolution of the hexagonal grid. Both interactions require the bin aggregates to be recomputed. In addition, users can hover the hexagons to receive details-on-demand and can set their preferred colors for the hexagons’ color scale. The application can be found online [cce23]. Note, however, that currently not all browsers support WebGPU yet.

3. WebGPU Implementation

The implementation consists of two main steps: pre-processing and visualization. These steps are illustrated in Figure 2.

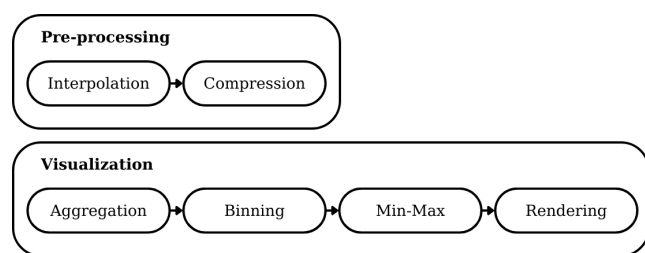


Figure 2: Overview of the offline pre-processing steps and the WebGPU compute and render pipeline steps.

In an offline pre-processing step using Python, we first impute missing values using a linear regression and then compress the data. This is necessary because the data is originally provided as CSV with 500MB [kag17]. Apart from removing unused columns like city names, we use delta encoding and discretization of temperature values within the reported uncertainty range to reduce the data

size. The data is then LZMA-compressed [LZM17]. Overall, we can achieve a compression rate of 99% compared to the original CSV-file.

On the client side, the compressed data is loaded into browser memory and LZMA-decompressed. For parallel filtering and aggregation, we set up three compute pipelines:

1. The **aggregation** compute pipeline filters and aggregates monthly temperature values according to the user-defined time periods for each city in parallel. It iterates through the city’s associated temperature data points, filtering entries that correspond to the specified time ranges, and subsequently computes the average temperature for both time ranges.
2. The **binning** compute pipeline evaluates the aggregated values for each hexagonal bin. It finds the cities within its geographic bounds and averages the temperature values of all contained cities for both given time periods. Finally, it computes the absolute difference between the two average temperature values. To effectively query the cities contained within the hexagon, they are organized in a KD-tree based on their geographic coordinates. The resulting grid buffer containing the average temperature differences and the number of evaluated temperature values is read back to the CPU for detail-on-demand tooltips.
3. The **min-max** compute pipeline is a parallel reduction step to determine the minimum and maximum temperature difference values across all bins. These are necessary to define the end points of the diverging color scale. This operation is performed per workgroup, i.e., across up to 32 hexagonal bins at once.

Finally, we set up two rendering pipelines: one for the world map in the background and another for the hexagon overlay. The latter computes each hexagon’s color from the bin’s absolute temperature difference value and the end points of the diverging color scale in the vertex shader.

4. Results and Conclusions

After changing the temperature ranges or the bin resolution, the three compute pipelines are sufficiently fast to maintain interactive frame rates. The three pipelines require an average overall computation time of <2ms (NVIDIA GeForce RTX 3050 Ti Laptop) and <10ms (Intel Iris Xe Graphics), respectively, after user interactions that require recomputation of aggregates. Decompression after receiving the data requires a few seconds when loading the page.

These first results show that WebGPU’s compute and render pipelines can significantly exceed the performance of state-of-the-art client-side solutions for aggregate visualizations, such as *Falcon* [MHH19], especially since no costly reindexing of data is necessary. We therefore see WebGPU as a promising new platform for interactive aggregate visualizations of millions of data points.

Acknowledgments

This work is partially supported by the Austrian Science Fund (FWF): P 36453. We thank Áron Samuel Kovács and Lukas Herzberger for discussions.

References

- [ber20] Berkeley earth climate data, 2020. URL: <https://berkeleyearth.org/data/>. 2
- [BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D³ data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2301–2309. doi:10.1109/TVCG.2011.185. 1
- [cce23] Climate Change Explorer—Online Demo, 2023. URL: <https://ccexplorer.github.io/>. 1, 2
- [cro15] Crossfilter—fast multidimensional filtering for coordinated views, 2015. URL: <https://crossfilter.github.io/crossfilter/>. 1
- [DPMO12] DUCHOWSKI A. T., PRICE M. M., MEYER M., ORERO P.: Aggregate gaze visualization with real-time heatmaps. In *Proceedings of the symposium on eye tracking research and applications* (2012), pp. 13–20. 1
- [DPU*22] DYKEN L., POUDEL P., USHER W., PETRUZZA S., CHEN J. Y., KUMAR S.: Graphwagu: Gpu powered large scale graph layout computation and rendering for the web. In *Eurographics Symposium on Parallel Graphics and Visualization* (2022). 1
- [EF09] ELMQVIST N., FEKETE J.-D.: Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE transactions on visualization and computer graphics* 16, 3 (2009), 439–454. 1
- [kag17] Climate change: Earth surface temperature data, 2017. URL: <https://www.kaggle.com/datasets/berkeleyearth/climate-change-earth-surface-temperature-data>. 2
- [LJH13] LIU Z., JIANG B., HEER J.: imMens: Real-time visual querying of big data. *Computer Graphics Forum* 32, 3pt4 (2013), 421–430. doi: <https://doi.org/10.1111/cgf.12129>. 1
- [LM19] LI J. K., MA K.-L.: P5: Portable progressive parallel processing pipelines for interactive data analysis and visualization. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 1151–1160. 1
- [LZM17] Lzma-js, 2017. URL: <https://github.com/LZMA-JS/LZMA-JS>. 2
- [MHH19] MORITZ D., HOWE B., HEER J.: Falcon: Balancing interactive latency and resolution sensitivity for scalable linked visualizations. In *Proceedings of the 2019 CHI conference on human factors in computing systems* (2019), pp. 1–11. 1, 2
- [SHCW22] STUMPFEGGER J., HÖHLEIN K., CRAIG G., WESTERMANN R.: Gpu accelerated scalable parallel coordinates plots. *Computers & Graphics* 109 (2022), 111–120. 1
- [SMWH17] SATYANARAYAN A., MORITZ D., WONGSUPHASAWAT K., HEER J.: Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 341–350. doi:10.1109/TVCG.2016.2599030. 1
- [UP20] USHER W., PASCUCCI V.: Interactive visualization of terascale data in the browser: Fact or fiction? In *2020 IEEE 10th Symposium on Large Data Analysis and Visualization (LDAV)* (2020), IEEE, pp. 27–36. 1