TimeVizBench: An Interactive Platform for Evaluating Techniques for Efficient Large Time Series Visualization

Vassilis Stamatopoulos*[1,2][0000-0002-9044-796X] bstam@athenarc.gr, Stavros Maroulis[1][0000-0003-2816-4368] stavmars@athenarc.gr, Christos Pantoleon[3][0000-0002-2312-5143] chr.pantoleon@aueb.gr, George Papastefanatos[1][0000-0002-9273-9843] gpapas@athenarc.gr, and Panos Vassiliadis[2][0000-0003-0085-6776] panos.vassiliadis@cs.uoi.gr

[1] ATHENA Research Center, Athens, Greece
[2] University of Ioannina, Ioannina, Greece
[3] Athens University of Economics and Business, Athens, Greece

Abstract. Interactive time series visualization is essential in domains like IoT monitoring but is often constrained by latency and scalability challenges. Various methods have been proposed to address these issues, each with different trade-offs between efficiency, interactivity, and visualization accuracy, making systematic evaluation crucial. Traditional benchmarking approaches, however, fail to capture user-perceived responsiveness and accuracy in real-world exploration scenarios. To bridge this gap, we introduce *TimeVizBench*, an interactive evaluation platform for scalable time series visualization methods across performance and accuracy dimensions. *TimeVizBench* enables users to configure different methods, explore visual outputs interactively, and dynamically assess performance and accuracy. It also provides a standardized interface for integrating and comparing additional methods.

Keywords: Interactive Visualization · Time Series · Approximate Visualization

1 Introduction

Visualizing large-scale, multivariate time series data presents unique challenges due to the sheer volume, high dimensionality, and dynamic nature of the data. In domains such as IoT monitoring, financial analysis, and anomaly detection, users rely on interactive visual exploration performing operations like panning, zooming, and pattern highlighting. Achieving interactive response times is essential for effective real-time analysis, while ensuring visualization accuracy is critical to avoid misleading or incomplete insights. However, the latency associated with fetching, processing, and rendering such data often hinders interactivity.

* Corresponding author: bstam@athenarc.gr

Approaches for Scalable Time Series Visualization. Various methods have been proposed to address the challenges of large-scale time series visualization. Traditional techniques, such as sampling and aggregation, reduce data volume to improve performance but often distort visual representation and affect visualization accuracy. Visualization accuracy is typically assessed at the pixel level rather than in the data domain, measuring how closely the rendered visualization aligns with the expected output from raw data. Metrics like the Structural Similarity Index Measure (SSIM) [12] quantify visual differences by evaluating structural and perceptual similarities, making them suitable for assessing visualization accuracy.

Visualization-aware methods, such as M4 [6], account for visualization parameters (e.g., width and height of the chart canvas) to ensure accurate representations. M4 aggregates data into pixel-wide intervals, preserving the visualization's shape with 100% accuracy. However, it requires querying all relevant data for each interaction, increasing latency and reducing interactivity.

Progressive approaches, like OM3 [11], incrementally refine visualizations through a precomputed multi-level representation, eventually achieving an error-free visualization. However, OM3 lacks error guarantees for intermediate visualizations, requiring users to wait until full convergence for accuracy assurances.

To improve interactivity, caching-based methods have emerged. MinMaxCache [9] reduces latency in time series visual exploration while ensuring accuracy. It dynamically aggregates and caches data at granularities optimized for visualization, considering user-defined accuracy constraints, enabling efficient reuse of cached results during panning and zooming. This approach minimizes redundant data fetching while maintaining pixel error-bound guarantees, ensuring accurate visual representations with low latency. Unlike precomputation-based methods, MinMaxCache adapts dynamically to user exploration and supports streaming data, making it well-suited for interactive scenarios.

Other methods, such as MinMaxLTTB [3], integrate min-max preselection with downsampling techniques like Largest Triangle Three Buckets (LTTB) to improve scalability while maintaining visual quality. However, they lack explicit error quantification relative to fully accurate references.

Challenge and Main Beneficiaries. Comparing and systematically evaluating these methods remains challenging. Existing evaluation practices often rely on controlled experiments or algorithm reimplementations, which are labor-intensive and fail to capture the dynamic, user-driven nature of modern visualization tools. Interactive visualization scenarios also demand careful consideration of user experiences—particularly perceptions of latency and accuracy—which directly influence both utility and usability. Addressing these needs requires an interactive experimentation platform that enables the evaluation of diverse methods across various metrics, facilitating the addition of new ones. Such a platform should enable its key beneficiaries, i.e., researchers, and software engineers to accomplish real-time exploration of trade-offs, systematically evaluate visualization techniques, and provide built-in support for measuring key metrics through a well-defined interface, ensuring ease of use and extensibility.

Related Work. While many techniques support interactive, low-latency, scalable visualization, the lack of standardized evaluation frameworks limits systematic comparison. The need for benchmarking in visualization has been well established [1], highlighting the importance of performance evaluation in interactive data systems. Generic benchmarks like TPC-H focus on backend performance metrics, while domain-specific benchmarks, such as those for time series data [7], provide specialized evaluation frameworks. While effective for assessing generic query performance and scalability, these benchmarks do not address the interactivity and accuracy requirements of visualization systems. Visualization-specific benchmarks, such as the Visual Analytics Benchmark Repository [10], focus on analytical effectiveness but offer limited support for evaluating system performance during interactive operations. More recent efforts include interactive data exploration benchmarks [2,4], which provide systematic approaches for gaining insights into query execution and system-level performance for interactive exploration and visualization. While these benchmarks enable standardized evaluation of system performance, they do not explicitly target timeseries visualization methods and often fail to capture the real-time user experience, including user-perceived latency, visual accuracy, and the critical trade-offs between usability and performance essential in interactive scenarios.

Contribution. In this work, we introduce *TimeVizBench*, an extensible platform for evaluating methods designed to enhance the interactivity and scalability of time series visual exploration, focusing on key dimensions such as performance and visualization accuracy. *TimeVizBench* supports state-of-the-art approaches like MinMaxCache [9] and M4 [6], while offering a flexible framework for integrating alternative algorithms. Method-specific parameters are declaratively defined and translated into user interface controls, enabling effortless experimentation. Users can interactively explore multivariate time series data through operations like pan and zoom, while comparing the efficiency and effectiveness of different methods in real time. The platform measures key performance metrics, including query time, network transfer, and rendering time, while also assessing visualization accuracy via SSIM and tracking data reduction efficiency based on the amount of retrieved data.

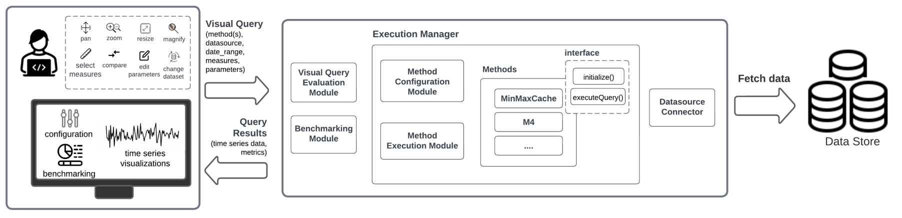


Fig. 1. *TimeVizBench* Platform Architecture

2 *TimeVizBench* Architecture

Figure 1 presents the architecture of the *TimeVizBench* system, designed for evaluating methods that support interactive, large-scale time series visualization. The platform is composed of a web-based UI and a backend API responsible for executing the evaluated methods and interacting with the database.

The UI allows users to select datasets, configure various parameters, and compare the visualizations generated by different methods. Each query specifies the method to execute, the datasource to query, the date range, measures to visualize, and any required parameters for the selected method. Additionally, many visualization-aware methods require parameters such as the width and height of the visualization canvas, which are automatically included. Query results include the data points (e.g., timestamps and values) to visualize for one or more variables, performance metrics (e.g., query time and total response time), and visualization quality metrics (e.g., SSIM).

The backend is orchestrated by the **Execution Manager**, which manages query evaluation, method execution, and benchmarking. To evaluate a new method, developers must implement a standardized interface and annotate it with `@VisualMethod`, giving it a name and a description. This annotation mechanism automatically registers the method with the system, making it available to both backend and frontend components.

Parameters for each method are defined using the `@Parameter` annotation. A built-in boolean field in the annotation setup, distinguishes **initialization parameters**, which are set once during method instantiation and **query-time parameters**, which are specified at runtime for each query (e.g., the accuracy constraint in MinMaxCache). When a method is initialized, any necessary precomputed data can be loaded within the `initialize()` routine.

When a user selects a method in the UI, the frontend dynamically queries the backend for its parameter definitions. These are extracted by introspecting the annotated Java fields, so the UI can render the appropriate configuration options without manual intervention. Outputs are always returned using standardized classes to ensure compatibility with frontend requirements.

The **Method Execution Module** processes the configured methods, generating and executing queries through the **Datasource Connector**. This connector retrieves data as needed from the specified datasource. Processed results, including time series data and computed metrics, are returned to the **Benchmarking Module**, which evaluates method performance across multiple dimensions. *TimeVizBench* reports metrics for three key evaluation dimensions:

- **Performance** is measured by *query time*, which captures the time taken by the database to evaluate the query, *network time* for data transfer, and *rendering time* for visualizing the results.
- **Accuracy** is assessed using the *Structural Similarity Index Measure (SSIM)*, quantifying how closely a method's visualization matches the reference visualization.

– **Data reduction efficiency** is evaluated by measuring the *amount of data retrieved from the database* for each method. This metric helps assess how effectively a method reduces the data volume needed to fetch from the data store, impacting both performance and visualization latency.

These metrics allow users to systematically compare methods and explore trade-offs between interactivity, accuracy, and data reduction efficiency.

Currently, the platform supports SQL databases and InfluxDB, with the **Datasource Connector** enabling metadata retrieval for available datasets and time series measures. The system is designed for extensibility, allowing new datasources to be easily added.

To ensure fair benchmarking, an optional cleaning method is invoked in the **Datasource Connector** after executing each query, which clears any cached results from the underlying database, minimizes the impact of database optimizations like caching and ensuring consistent evaluation conditions across methods. The processed results and benchmarking metrics are sent to the frontend, enabling users to analyze and compare methods across latency, accuracy, and interactivity dimensions.

Implementation Details. *TimeVizBench* features a Java 17 backend built with Spring Boot and a React-Redux frontend. Time series rendering is handled using D3.js. The source code is available under the MIT license[4].

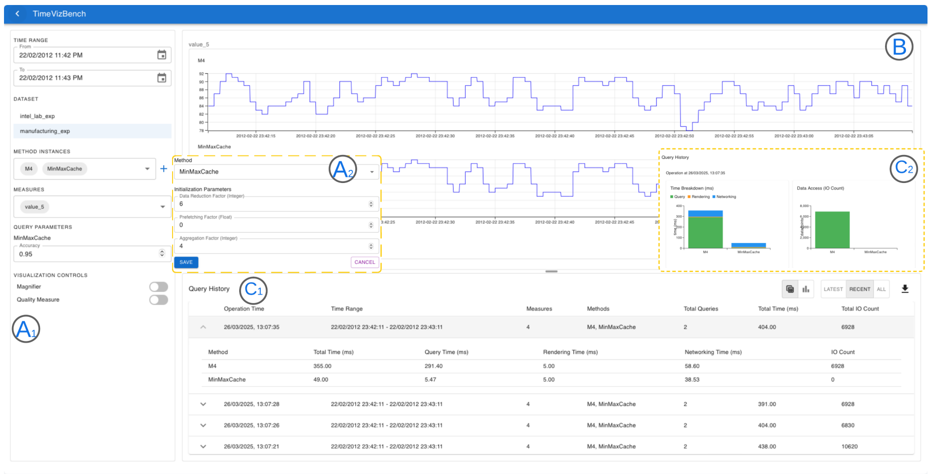
3 *TimeVizBench* User Interface

This section presents the *TimeVizBench* interface (Fig. 2), which comprises three components: the *Configuration Panel*, *Visualization Panel*, and *Performance Metrics Panel*.

The *Control Panel* ($\textcircled{\mathbf{A}_1}$), enables users to set up the parameters for exploration. Users can select a time range through interactive date and time pickers, though this interval can also be adjusted dynamically via pan and zoom on the charts. Additionally, users can choose from available datasets and specify one or more measures to visualize. The interface supports multi-value selection, enabling analysis of multiple variables simultaneously.

Users can also select different methods for evaluation. Upon selection of a method, users are prompted to configure the *initialization parameters*, which remain fixed for that method instance throughout the session ($\textcircled{\mathbf{A}_2}$). In contrast, *query-time parameters*, such as accuracy thresholds, can be modified dynamically at any time, even after instantiation. The platform also allows for comparing multiple instances of the same method by configuring different values for either the initialization or query-time parameters. This flexibility enables users to systematically compare methods or explore variations of a single method under different configurations.

[4] The source code is available at *https://github.com/athenarc/TimeVizBench*



TIME RANGE
From: 22/02/2012 11:42 PM
To: 22/02/2012 11:43 PM

DATASET
intel_lab_exp
manufacturing_exp

METHOD INSTANCES
M4 MinMaxCache

MEASURES
value_5

QUERY PARAMETERS
MinMaxCache
Accuracy: 0.95

VISUALIZATION CONTROLS
Magnifier
Quality Measure

Method: MinMaxCache
Initialization Parameters
Data Reduction Factor (Integer): 6
Prefetching Factor (Float): 0
Aggregation Factor (Integer): 4

Query History

Operation Time	Time Range	Measures	Methods	Total Queries	Total Time (ms)	Total IO Count
26/03/2025, 13:07:35	22/02/2012 23:42:11 - 22/02/2012 23:43:11	4	M4, MinMaxCache	2	404.00	6928

Method	Total Time (ms)	Query Time (ms)	Rendering Time (ms)	Networking Time (ms)	IO Count
M4	355.00	291.40	5.00	58.60	6928
MinMaxCache	49.00	5.47	5.00	38.53	0

Operation Time	Time Range	Measures	Methods	Total Queries	Total Time (ms)	Total IO Count
26/03/2025, 13:07:28	22/02/2012 23:42:11 - 22/02/2012 23:43:11	4	M4, MinMaxCache	2	391.00	6928
26/03/2025, 13:07:26	22/02/2012 23:42:11 - 22/02/2012 23:43:11	4	M4, MinMaxCache	2	404.00	6830
26/03/2025, 13:07:21	22/02/2012 23:42:11 - 22/02/2012 23:43:11	4	M4, MinMaxCache	2	438.00	10620

Fig. 2. *TimeVizBench* User Interface for the interactive evaluation of methods for large-scale time series exploration. The *Control Panel* ($\mathbf{A}_1$), enables users to set up the parameters for exploration. Upon selection of a method, users are prompted to configure the *initialization parameters* ($\mathbf{A}_2$). The *Visualization Panel* (**B**) displays time series line charts for the selected measures and method instances. The *Performance Metrics Panel* (***C***) *provides insights into the performance of each configured method and is organized into two complementary views. A* table view ($\boldsymbol{C}_1$) *and a* chart view ($\boldsymbol{C}_2$).

Furthermore, this panel provides controls for the generated visualization. When enabled, the *Magnifier*, displays a circled segment of the time series in an enlarged overlay for closer inspection. Meanwhile, the *Quality Measure* switch overlays an error-free reference series on each chart for comparison against method-generated visualizations and displays the corresponding Structural Similarity Index Measure (SSIM) [12]. Due to the large-scale nature of the data, fetching raw data to verify accuracy would be prohibitively expensive; however, M4 has been verified as error-free (SSIM = 1) relative to the actual raw data [6], making it an ideal reference for measuring how closely each method's output matches the baseline.

The *Visualization Panel* (B) displays time series line charts for the selected measures and method instances. Each measure is shown in a separate panel, with visualizations for all configured methods stacked vertically for direct comparison. The visualizations are synchronized, ensuring that user interactions such as panning or zooming on one chart automatically update all others.

The *Performance Metrics Panel* (C), provides insights into the performance of each configured method and is organized into two complementary views. A *table view* ($\mathbf{C_1}$) lists each query with its time interval, chosen dataset, requested measures, and all methods used, grouping together the performance metrics (e.g., query time, network time, rendering time, and data I/O counts). A *chart view* ($\mathbf{C_2}$) displays the *same* metrics in bar-chart form, with each bar broken down to show the contributions of querying, networking, and rendering, alongside a visualization of data retrieval counts.

Additionally, *TimeVizBench* provides an export functionality, allowing users to download query history and corresponding performance metrics in CSV format. Each exported file includes details such as the time interval, selected dataset, measures requested, and performance metrics recorded for each method. This feature facilitates further offline analysis, enabling users to systematically compare different configurations, track performance trends over time, and reproduce evaluations for consistency.

Availability. The tool and its functionalities can be accessed online[5]. A video demonstration is also available[6].

4 Demonstration Outline

In this section, we describe the demonstration scenario for *TimeVizBench*, where attendees will explore methods for scalable, low-latency time series visualization. The demonstration showcases *TimeVizBench*'s ability to benchmark visualization methods across performance and accuracy dimensions using real-world datasets. Attendees will be asked to perform the following tasks:

– Add an example method for benchmarking, implemented via *TimeVizBench*'s interface (e.g., a simple averaging method per pixel column).
– Select a dataset from preloaded time series data (e.g., sensor readings from [8] or electrical power measurements from [5]) and choose one or more measures for visualization.
– Instantiate and configure visualization methods, with support for multiple configurations of the same method.
– Interactively explore time series visualizations, with synchronized panning and zooming across selected methods for direct comparison.
– Analyze performance metrics, including query time, total response time, and the amount of data retrieved from the database per method.
– Compare visualization accuracy using SSIM to assess differences between method-generated visualizations and a reference visualization.
– Export session results, including query history, method configurations, and performance metrics, for further analysis.

By engaging with these tasks, attendees will gain hands-on experience with *TimeVizBench*, understanding how different methods balance interactivity, performance, and accuracy in large-scale time series visualization.

Acknowledgments

This work was supported by the ExtremeXP project (EU Horizon program, GA 101093164).

[5] *http://timevizbench.imsi.athenarc.gr/*
[6] *https://vimeo.com/1070287190*

References

1. Battle, L., Chang, R., Heer, J., Stonebraker, M.: Position statement: The case for a visualization performance benchmark. In: 2017 IEEE Workshop on Data Systems for Interactive Analysis (DSIA). pp. 1–5 (2017). https://doi.org/10.1109/DSIA.2017.8339089
2. Battle, L., Eichmann, P., Angelini, M., Catarci, T., Santucci, G., Zheng, Y., Binnig, C., Fekete, J.D., Moritz, D.: Database benchmarking for supporting real-time interactive querying of large data. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. p. 1571–1587. SIGMOD '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3318464.3389732, `https://doi.org/10.1145/3318464.3389732`
3. Donckt, J.V.D., Donckt, J.V.D., Rademaker, M., Hoecke, S.V.: Minmaxlttb: Leveraging minmax-preselection to scale lttb (2023), `https://arxiv.org/abs/2305.00332`
4. Eichmann, P., Zgraggen, E., Binnig, C., Kraska, T.: Idebench: A benchmark for interactive data exploration. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. p. 1555–1569. SIGMOD '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3318464.3380574, `https://doi.org/10.1145/3318464.3380574`
5. Jerzak, Z., Heinze, T., Fehr, M., Grober, D., Hartung, R., Stojanovic, N.: The debs 2012 grand challenge. DEBS pp. 393–398 (2012), `https://debs.org/grand-challenges/2012/`
6. Jugel, U., Jerzak, Z., Hackenbroich, G., Markl, V.: M4: a visualization-oriented time series data aggregation. Proceedings of the VLDB Endowment **7**(10), 797–808 (2014)
7. Khelifati, A., Khayati, M., Dignös, A., Difallah, D., Cudré-Mauroux, P.: Tsm-bench: Benchmarking time series database systems for monitoring applications. Proc. VLDB Endow. **16**(11), 3363–3376 (Jul 2023). https://doi.org/10.14778/3611479.3611532, `https://doi.org/10.14778/3611479.3611532`
8. Lab, I.B.R.: Intel lab dataset (2004), `http://db.csail.mit.edu/labdata/labdata.html`
9. Maroulis, S., Stamatopoulos, V., Papastefanatos, G., Terrovitis, M.: Visualization-aware time series min-max caching with error bound guarantees. Proc. VLDB Endow. **17**(8), 2091–2103 (Apr 2024). https://doi.org/10.14778/3659437.3659460, `https://doi.org/10.14778/3659437.3659460`
10. Plaisant, C., Fekete, J.D., Grinstein, G.: Promoting insight-based evaluation of visualizations: From contest to benchmark repository. IEEE Transactions on Visualization and Computer Graphics **14**(1), 120–134 (2008). https://doi.org/10.1109/TVCG.2007.70412
11. Wang, Y., Wang, Y., Chen, X., Zhao, Y., Zhang, F., Wu, E., Fu, C.W., Yu, X.: Om3: An ordered multi-level min-max representation for interactive progressive visualization of time series. Proceedings of the ACM on Management of Data **1**(2), 1–24 (2023)
12. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)