Toward an Interaction-Driven Framework for Modeling Big Data Visualization Systems

D. Benvenuti[1], G. Fiordeponti[1], H. Cheng[2], T. Catarci[1], J.D. Fekete[3], G. Santucci[1], M. Angelini[1] and L. Battle[4]

[1] Department of Computer, Control, and Management Engineering Antonio Ruberti, University of Rome "La Sapienza", Rome, Italy
[2] University of Maryland, College Park, Maryland, United States
[3] Aviz, Inria, Saclay, France
[4] Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, Washington, United States

Abstract

Designing big data visualization applications is challenging due to their complex yet isolated development. One of the most common issues is an increase in latency that can be experienced while interacting with the system. There exists a variety of optimization techniques to handle this issue in specific scenarios, but we lack models for integrating them in a holistic way, hindering the integration of complementary functionality and hampering consistent evaluation across systems. In response, we present a framework for modeling the big data visualization pipeline which builds a bridge between the Visualization, Human-Computer Interaction, and Database communities by integrating their individual contributions within a single, easily interpretable pipeline. With this framework, visualization applications can become aware of the full end-to-end context, making it easier to determine which subset of optimizations best suits the current context.

CCS Concepts

• ***Information systems*** → ***Information systems applications;*** • ***Human-centered computing*** → ***Human computer interaction (HCI);***

1. Introduction

When users interact with large data in a visualization application, its response time is crucial in keeping users efficient and engaged, particularly for exploratory data analysis (EDA). Even latencies as low as 500 ms can be detrimental [LH14, WK09]. Latencies can be introduced in each of the three layers of the visualization pipeline: *(i)* data layer; *(ii)* interaction layer and *(iii)* rendering layer. For example, for large data and complex queries, even commercial database systems (DBMS) may fail to meet the performance needs of interactive visual analysis systems [BEA*20], introducing latencies in the data layer. In practice, most visualization applications are designed for small data or resort to ad-hoc mechanisms to try to meet the latency constraints on larger data (e.g., Spotfire [Spo95]) by optimizing only one of the three layers. Although we have good models for optimizing the data and rendering layers (e.g., BIRCH [ZRL96], DEVise [LRB*97] etc.), we lack one for the interaction layer. Furthermore, no formal framework exists to connect known performance models in the visualization and HCI communities with optimization strategies at the data management level. To fill this gap between these communities we present a framework for modeling how user interactions affect each component of the data visualization pipeline. The main idea of the framework is to introduce a translation layer and to model interactions using augmented *statecharts* [Har87]. Users' interactions are captured and then translated from low-level user actions into high-level database queries and rendering information that are annotated in the statechart. The annotated statechart can be explored by the designers to understand which interactions suffer from excessive latency and how to fix it. By applying this framework, it becomes possible to: *(i)* have a complete model of how users interact with the visualization application that can be annotated with information useful for optimization (e.g., latency constraints, probabilities); *(ii)* know in advance the SQL queries that could be triggered from the current interaction state and *(iii)* derive through them the optimized computations needed to (re)render the results.

2. Framework

The proposed framework, visible in Figure 1, is built on the classic visualization pipeline, modeled into three different layers: *(i)* a *data layer*, which collects the relevant data through selection and applies optional binning and aggregation operations; *(ii)* a *rendering layer* that uses the binned and aggregated data to generate visualization components and computes the extent of each rendered column to produce axes and scales and *(iii)* an *interaction layer* in charge of issuing queries either from dynamic query widgets or direct manipulation of the visualization components. Big data exploration is generally made possible through the cooperation of these layers, but each application is designed to address separate parts of the pipeline. Without a connecting framework, it becomes difficult to

DOI: 10.2312/evp.20221125

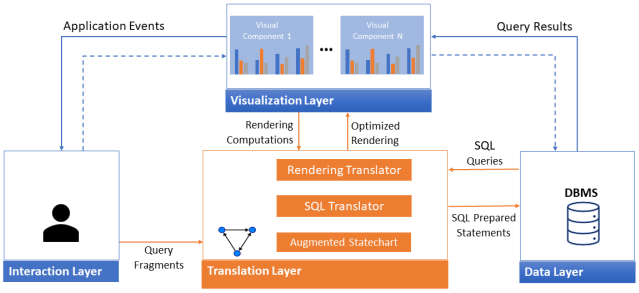

Figure 1: *The interaction-driven framework for big data exploration (the translation layer is highlighted in orange).*

Figure 2: *A snapshot of the dashboard obtained applying the Interaction-Driven Framework to a crossfilter interface. The user is currently brushing on the "Distance in Miles" histogram.*

understand how a new DBMS optimization strategy applies to new and existing visualization interface designs. As a result, optimizations developed in the database community run the risk of being point solutions at best, and irrelevant at worst in solving the performance issues observed by the visualization community.

One of the key ideas behind this framework is the addition of a *translation layer* to integrate critical information from the other three. It *(i)* receives query fragments (from the user's activity) from the interaction layer, *(ii)* combines and transforms them, adding meaningful information about the context in which the interaction happened and *(iii)* translates them into SQL queries and rendering computations. By specifying interactions as state machines with latency information and leveraging the translation layer, our unified framework enables more versatile management of query load driven by the user's actions and provides a centralized layer of computation that allows us to exploit, rather than modify, the existing interaction and data management layers. We envision this interaction-driven framework to be supported, based on its functionalities, by three different components: *(i)* Augmented Statechart, *(ii)* SQL Translator and *(iii)* Rendering Translator. In the following, we focus on the ongoing work for the first two components.

(i) We define an *atomic interaction* as a statechart containing the minimum states and transitions needed to represent an interaction, which is typically one state and three transitions for stateful actions or two transitions only for stateless ones. Examples of atomic interactions are: hover, click, zoom, drag, pan and brush. We can combine atomic interactions to model more complex interactions (e.g., "hover" and "click" to model the interaction with a button).

*(ii)*A user interaction is captured by the *interaction layer* and sent to the *translation layer* which extracts the SQL queries to be executed. This feature is implemented through the *Label Module* (*LM*), which gives the result back by labeling the statechart. The generations of SQL prepared statement is implemented by the *Query Module* (*QM*) which manages a *Query Storage* (*QS*), i.e. a data structure containing SQL Fragments. When the statechart is initialized, *LM* assigns a specific set of query parameters to each of its transitions that contribute to the SQL Statements. Those parameters are referred to as *Query Metadata*, organized in a set of *query labels*. When a transition leads to a state containing a Query Metadata, that information is acquired by *QM*: for each Query Label, a SQL Prepared Statement is created by taking from *QS* all the different fragments and filled with dimensional values and parameters.

3. Usage-Scenario

We started experimenting with the proposed framework on a demanding yet popular visualization application, crossfilter [MHH19a]. We modeled the interactions by combining the JSON definitions of *atomic interactions* and running the statechart using the *XState* JavaScript library [XSt22], which is used to build the main components of the *translation layer*. When a user's action is captured by a listener, we pass the information to *Xstate* which updates the statechart accordingly and labels it with query fragments. Finally, we created a dashboard to show in real-time: *(i)* each state reachable in *n* steps from the current one and *(ii)* each query fragment that can be generated in those steps. A snapshot of this dashboard can be seen in Figure 2 in which we captured its state while the user is idling while brushing on the "Distance in Miles" histogram. The *interaction* layer collects information about the current state in the statechart, the last transition triggered by the user, the list of two steps paths branching from the current state (e.g., we can reach the "inactive" state if the "MOUSELEAVE" event occurs) and metadata about the context in which the interaction is happening (e.g., the active view and the values of the current brush). The *translation layer* is updated in real-time and generates the Query Labels (see section 2), visible in the "Active Meta" field of the dashboard.

4. Conclusions

The proposed interaction-driven framework can enable a higher awareness, both at design and execution time, of the existing interrelations among the four layers considered (data, interaction, rendering, and the newly introduced translation). It can accommodate also different optimization strategies like all the ones listed in Battle and Scheidegger work [BS20]. As examples, we cite **Multi-Query Optimization** (e.g., SeeDB [VRM*15], Zenvisage [SKL*16], and Voyager [WMA*16]) producing many simultaneous queries; , **Materialized Views** (e.g., Falcon [MHH19b], imMens [LJH13], and Nanocubes [LKS13]), where the key challenge lies in identifying which queries to actually materialize; **User Modeling** (e.g., ForeCache [BCS16]) where the key challenge is to develop an accurate model of user interaction behavior used to predict future interactions.

References

[BCS16] BATTLE L., CHANG R., STONEBRAKER M.: Dynamic Prefetching of Data Tiles for Interactive Visualization. In *Proceedings of the 2016 International Conference on Management of Data* (New York, NY, USA, 2016), SIGMOD '16, ACM, pp. 1363–1375. URL: http://doi.acm.org/10.1145/2882903.2882919, doi:10.1145/2882903.2882919. 2

[BEA*20] 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* (New York, NY, USA, June 2020), SIGMOD '20, Association for Computing Machinery, pp. 1571–1587. URL: https://doi.org/10.1145/3318464.3389732, doi:10.1145/3318464.3389732. 1

[BS20] BATTLE L., SCHEIDEGGER C.: A structured review of data management technology for interactive visualization and analysis. *IEEE Trans. Vis. Comput. Graphics* (2020), 1–1. doi:10.1109/TVCG.2020.3028891. 2

[Har87] HAREL D.: Statecharts: a visual formalism for complex systems. *Science of Computer Programming 8*, 3 (1987), 231–274. URL: https://www.sciencedirect.com/science/article/pii/0167642387900359, doi:https://doi.org/10.1016/0167-6423(87)90035-9. 1

[LH14] LIU Z., HEER J.: The Effects of Interactive Latency on Exploratory Visual Analysis. *IEEE Trans. Vis. Comput. Graphics* (2014). URL: http://idl.cs.washington.edu/papers/latency. 1

[LJH13] LIU Z., JIANG B., HEER J.: imMens: Real-time Visual Querying of Big Data. *Computer Graphics Forum 32*, 3pt4 (June 2013), 421–430. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12129, doi:10.1111/cgf.12129. 2

[LKS13] LINS L., KLOSOWSKI J. T., SCHEIDEGGER C.: Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. *IEEE Trans. Vis. Comput. Graphics 19*, 12 (Dec. 2013), 2456–2465. doi:10.1109/TVCG.2013.179. 2

[LRB*97] LIVNY M., RAMAKRISHNAN R., BEYER K., CHEN G., DONJERKOVIC D., LAWANDE S., MYLLYMAKI J., WENGER K.: Devise: Integrated querying and visual exploration of large datasets. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1997), SIGMOD '97, Association for Computing Machinery, p. 301–312. URL: https://doi.org/10.1145/253260.253335, doi:10.1145/253260.253335. 1

[MHH19a] MORITZ D., HOWE B., HEER J.: Falcon: Balancing Interactive Latency and Resolution Sensitivity for Scalable Linked Visualizations. In *Proc. SIGCHI Conf. Human Factors Comp. Sys.* (New York, NY, USA, 2019), CHI '19, ACM. URL: https://doi.org/10.1145/3290605.3300924, doi:10.1145/3290605.3300924. 2

[MHH19b] MORITZ D., HOWE B., HEER J.: *Falcon: Balancing Interactive Latency and Resolution Sensitivity for Scalable Linked Visualizations*. Association for Computing Machinery, New York, NY, USA, 2019, p. 1–11. URL: https://doi.org/10.1145/3290605.3300924. 2

[SKL*16] SIDDIQUI T., KIM A., LEE J., KARAHALIOS K., PARAMESWARAN A.: Effortless data exploration with zenvisage: An expressive and interactive visual analytics system. *Proceedings of the VLDB Endowment 10*, 4 (2016). 2

[Spo95] Tico spotfire. https://www.tibco.com/products/tibco-spotfire, 1995. Accessed: 2019-07-20. 1

[VRM*15] VARTAK M., RAHMAN S., MADDEN S., PARAMESWARAN A., POLYZOTIS N.: Seedb: Efficient data-driven visualization recommendations to support visual analytics. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases* (2015), vol. 8, NIH Public Access, p. 2182. 2

[WK09] WALOSZEK G., KREICHGAUER U.: User-centered evaluation of the responsiveness of applications. In *IFIP Conference on Human-Computer Interaction* (2009), Springer, pp. 239–242. 1

[WMA*16] WONGSUPHASAWAT K., MORITZ D., ANAND A., MACKINLAY J., HOWE B., HEER J.: Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 649–658. doi:10.1109/TVCG.2015.2467191. 2

[XSt22] XState documentation. https://xstate.js.org/docs/, 2022. [Online; accessed 11-April-2022]. 2

[ZRL96] ZHANG T., RAMAKRISHNAN R., LIVNY M.: Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1996), SIGMOD '96, Association for Computing Machinery, p. 103–114. URL: https://doi.org/10.1145/233269.233324, doi:10.1145/233269.233324. 1