Journal of Big Data Analytics in Transportation (2019) 1:147–159
https://doi.org/10.1007/s42421-019-00010-y

ORIGINAL PAPER

GPU-Enabled Visual Analytics Framework for Big Transportation Datasets

Yaw Adu-Gyamfi[1]

Received: 1 July 2019 / Revised: 23 September 2019 / Accepted: 10 October 2019 / Published online: 24 October 2019

Abstract
Transportation agencies rely on a variety of data sources for condition monitoring of their assets and making critical decisions such as infrastructure investments and project prioritization. Recent exponential increase in the volumes of these datasets has been causing significant information overload problems for data analysts; data curation process has increasingly become time consuming as legacy CPU-based systems are reaching their limits for processing and visualizing relevant trends in these massive datasets. There is a need for new tools that can consume these new datasets and provide analytics at rates resonant with the speed of human thought. The current paper proposes a new framework that allows for both multidimensional visualization and analytics to be carried seamlessly on large transportation datasets. The framework stores data in a massively parallel database and leverages the immense computational power available in graphical processing units (GPUs) to carry out data analytics and rendering on the fly via a Structured Query Language which interacts with the underlying GPU database. A front-end is designed for near-instant rendering of queried results on simple charts and maps to enable decision makers to drill down insights quickly. The framework is used to develop applications for analyzing big transportation datasets with over 100 million rows. Performance benchmarking experiments conducted showed that the methodology developed is able to provide real-time visual updates for big data in less than 100 ms. The performance of the developed framework was also compared with CPU-based visual analytics platforms such as Tableau and D3.

Keywords Big data analytics · Graphical processing units · Interactive visualiztion

Introduction

Visual analytics involves three main aspects: visualization, interactivity and analytics. Whereas visualization provides a meaningful display of data through charts and maps, interactivity enables users to explore data, ask different questions and find trends which may lead to new knowledge. Analytics on the other hand performs computations, aggregations and data reductions. Traditional transportation data processing pipelines treat visualization interactivity and analytics as two distinct components. The reason for separating analytics from visualization interactivity is due to the fact that web browsers, although have considerably improved in their ability to render objects quickly, have very low computing capacity in the face of big data. Data computations are therefore carried out with high-performance clusters and super computers, whereas visual interactions are carried out on the browser. The separation of both components has, however, created bottlenecks in the data curation process, which tend to impede the seamless flow of information for discovering new insights from data.

Transportation agencies are increasingly utilizing visual analytics as part of the data curation process to explore infinite paths of the "whats," and "whys" behind their data. Visual analytics enables them to generate different views of data through a dynamic and iterative process for answering questions, identifying problems and making unexpected discoveries (Nancy 2018). For visual analytics to be effective, the view of the data should update immediately with each visual query. Heer and Shneiderman (2012) postulated that an interactive, visual analytic system must be able to respond to queries at rates resonant with the pace of human thought. This will mean that the response rates for visual systems should be not more than 0.1 s. A user's flow of thought is interrupted and is likely to lose the feeling

✉ Yaw Adu-Gyamfi
adugyamfiy@missouri.edu

[1] Department of Civil and Environmental Engineering, University of Missouri, Columbia, USA

of operating directly on data if it takes more than 1 s to respond. For response delays longer than 10 s, users may want to perform other tasks while waiting for the system to respond. Valerie and Denis (2014) referred to this as the three categories of responsiveness (0.1, 1 and 10 s).

ArcGIS, Tableau and D3 are arguably the predominant visual analytic platforms used by most transportation agencies. The NHTSA (National Highway Traffic Safety Administration), for example, uses Tableau, an analytical visualization tool to reveal insights into speed related traffic fatalities across the USA (NHTSA 2016). Other agencies such as Virginia Department of Transportation (VDOT 2015), Bureau of Transportation Statistics (BTS) (2019) and Iowa Department of Transportation (Adu-Gyamfi et al. 2016; IOWADOT 2018) use similar platforms for drilling into the work zone, traffic and freight data, respectively. The size of data being visualized on these platforms ranges between several megabytes to a few gigabytes. Significant latencies can be observed in view of updates when the size of data being visualized exceeds 250 megabytes.

For relatively large datasets (5 GB or more), it is challenging, if not impossible, to achieve real-time visual updates with conventional visual analytic platforms. Recent developments aimed at handling big transportation data leverages high-performance computing clusters in the back end for all the heavy-lifting computations including data ingestion, aggregation, integration and reduction (Badu-Marfo et al. 2019; Islam and Sharma 2019). The filtered, aggregated and lightweight data are subsequently pushed to the front end for visual exploration. Although this approach provides a practical means for taming the "burden" of big data, it limits the power of visual analytics as fine details are lost through a series of aggregation and filtering processes. The goal of this paper is to develop a framework that enables visualization, interactivity and analytics of big datasets in the browser. The framework utilizes graphical processing units (GPUs) to enable heavy-lifting computations such as data reduction, aggregation and filtering to be carried out with user interactions from the front end.

The remainder of this paper is organized as follows: first, we highlight related research and recent data visualization trends in transportation. Next, the design framework including the key components of the visual analytic platform developed are explained. This section will also discuss the database architecture and data processing pipeline used to facilitate visualization of big datasets in the browser. The following section will highlight the transportation visualization example applications developed using our framework. In later sections, we develop performance benchmarks for the methodology and compare it to conventional techniques for visualizing transportation datasets. Conclusions and recommendations for future research are made in the last section.

Visual Analytic Trends in Transportation

The challenges of big data are driving transportation agencies to explore new and effective methods of data visualization that leads to actionable insights for transportation systems operation and management. Several visual analytic pipelines have been developed to help overcome some of the challenges in areas such as traffic operations, incident management and transit performance monitoring (Brennan et al. 2019; Chen et al. 2015a; Sharma et al. 2017).

Picozzi et al. (2013), for example, used an off-line processing engine to store yearly traffic crash information in a simple JSON format and precomputed spatiotemporal features including crash frequency by location, average traffic volumes per road segment, etc. The JSON files were later integrated into an online processing engine which provided an interactive visualization of the crash data by using charts, maps and heatmaps developed using D3 Javascript library. This pipeline provides significantly high levels of interactivity for the user. Different views of the crash database can be explored interactively on the fly, giving users the flexibility to answer different questions about the data. A key limitation of this approach is its inability to visualize large datasets. Significant latencies are observed when the size of the JSON database exceeds 200 megabytes. Utilizing a much scalable database like MongoDB to store the data could reduce these latencies.

In the area of transit, Abdullah et al. (2017) developed a Web-based visualization for transit operation and performance monitoring. The tool utilized MongoDB, a NoSQL database to store bus trajectory data, precomputed performance measures and then used an online GIS tool to visualize output results. A key limitation of the pipeline adopted by the authors is its inability to capture multidimensional views of the data being visualized: single charts or images typically provide answers to a handful of questions. In addition, although users could interact with the data via filtering and aggregation tools on the front end, the charts produced had limited interactivity. This could potentially limit users' ability to drill down the data and discovery patterns. Other variants of this visualization pipeline have been proposed in Chen et al. (2015b) and Sobral et al. (2019). Andrienko et al. (2017) explored the use of the space time cube (STC) to visualize highly complex, multidimensional data. In their proposed visualization framework, STC is used to represent both spatial and temporal aspects of vehicle trajectory and associated events such as delays and crashes, in a single chart. The interactivity of this visualization method is, however, limited to only zooming and panning operations. The tool can generate different views of the visualized data; however, it

is unable to handle on-the-fly computations. Because STCs use a single chart to visualize the different dimensions of the data, they have a tendency to overload the user with information.

Data Visualization with GPUs

The use of GPUs for scientific computing and visualization is not new (Mi et al. 2016; Liu et al. 2013; Moritz et al. 2019; Mostak 2016). There are two main features that make GPUs very attractive for handling big data. First, compared to CPUs, they have many more cores with much finer levels of parallelism for carrying out compute-intensive tasks. For example, a typical graphics card today includes up to several thousands of cores. Second, GPUs have a high memory bandwidth, which enables them to access data at a speed of about 100 GB per second. This feature is particularly relevant for low-latency rendering or visualization of big data. In spite of these features, GPU-based data visualization suffers some drawbacks, which have led to low adoption rates over the years. One of the key drawbacks of GPUs is its memory. GPU memory is often limited compared to CPU. Until recently, high-end GPUs could only boast of up to 6 GB RAM compared to 64–128 GB RAM on board CPUs. Although GPU RAM has improved with the introduction of P100s and V100s, they come at a steep price compared to the memory of CPU systems. A second drawback of GPUs is the low data transfer rate from CPU to GPU and vice versa. Although this drawback is still persistent, the development of Peripheral Component Interconnect (PCIs) bus has significantly improved the speed to about 12 GB per second.

Different GPU-based architectures have been explored for large data visualization. Mi et al. (2016) proposed a full-blown, GPU-centric design for exploring large time series and multidimensional datasets. In their design, both data storage and processing are handled in the GPU memory. The CPU is only used to generate user interactions or queries. By avoiding data transfers from GPU to CPU, and leveraging parallel processing for data aggregation and reduction, the authors were able to process and visualize billions of time series records at very low latencies. Liu et al. (2013) also developed "imMens", a browser-based visual analysis system which utilized WebGL for both data processing and rendering in the GPU. They achieved significantly high processing speeds by using data reduction strategies such as binned aggregation and sampling to process billions of records at a sustained 50 frames per second brushing and interactivity. Moritz et al. (2019), designed "Falcon", a client-GPU-based visualization platform designed for super-fast rendering of big data. It achieved state-of-the-art big data processing and rendering speeds by making principled trade-offs between latency and resolution. The client is designed to handle up to a million records with no latencies. For larger datasets, processing is off-loaded to a GPU database system.

Design Framework

Most GPU-based visualization frameworks are designed with the assumption that the GPU has enough memory capacity to consume all the data being processed. As a result, such designs do not have a systematic way of dealing with datasets which are bigger than the GPU memory. Their general performance degrades exponentially when their limit is reached. Taking this limitation into consideration, our visual analytics framework leverages a hybrid CPU–GPU architecture which optimizes the use of GPU memory by leveraging a cluster of CPUs to efficiently store and process part of the data when GPU memory capacity is overutilized. Our visualization framework is supported by OmniSci Core, a *massively parallel database* (*MapD*) system used for in-memory GPU data storage and processing (Mostak 2014). MapD first splits row fragments of a data table into constituent columns. Each column is then written to an appropriate chunk. Chunks are transferred to GPU when full to avoid memory overhead. For data processing, all requests are pushed through a query optimizer which determines the quickest way to execute the query, finds appropriate compiled GPU code and then executes code to process data. Results are compressed into bitmaps and transferred from GPU to CPU over PCI for visualization.

Our visualization framework has two main aspects: (1) hybrid CPU/GPU database for storing data and (2) data processing and rendering engine on CPU/GPU. Figure 1 shows the architecture of the visual analytics platform. The main benefit of our design over conventional techniques is that by leveraging the parallel architecture of GPU and CPU clusters for data storage and processing, we are able to aggregate and visualize big datasets on demand instead of precomputing.

CPU–GPU Storage Database

Due to the limitations of GPU memory, we adopted a CPU–GPU architecture for storing data. On the GPU, we leveraged a column-oriented relational database that stores data in columns instead of rows. A decision tree matrix shown in Fig. 2 is used to determine which columns in a database stay in GPU memory and which ones are moved into CPU memory. In general, columns that are frequently accessed by a user are kept in the GPU. Other columns with geospatial information such as latitude–longitudes and timestamps are also ranked as high-priority columns for GPU in-memory storage. On the CPU, a Cassandra (also a column store) database cluster is used to store columns

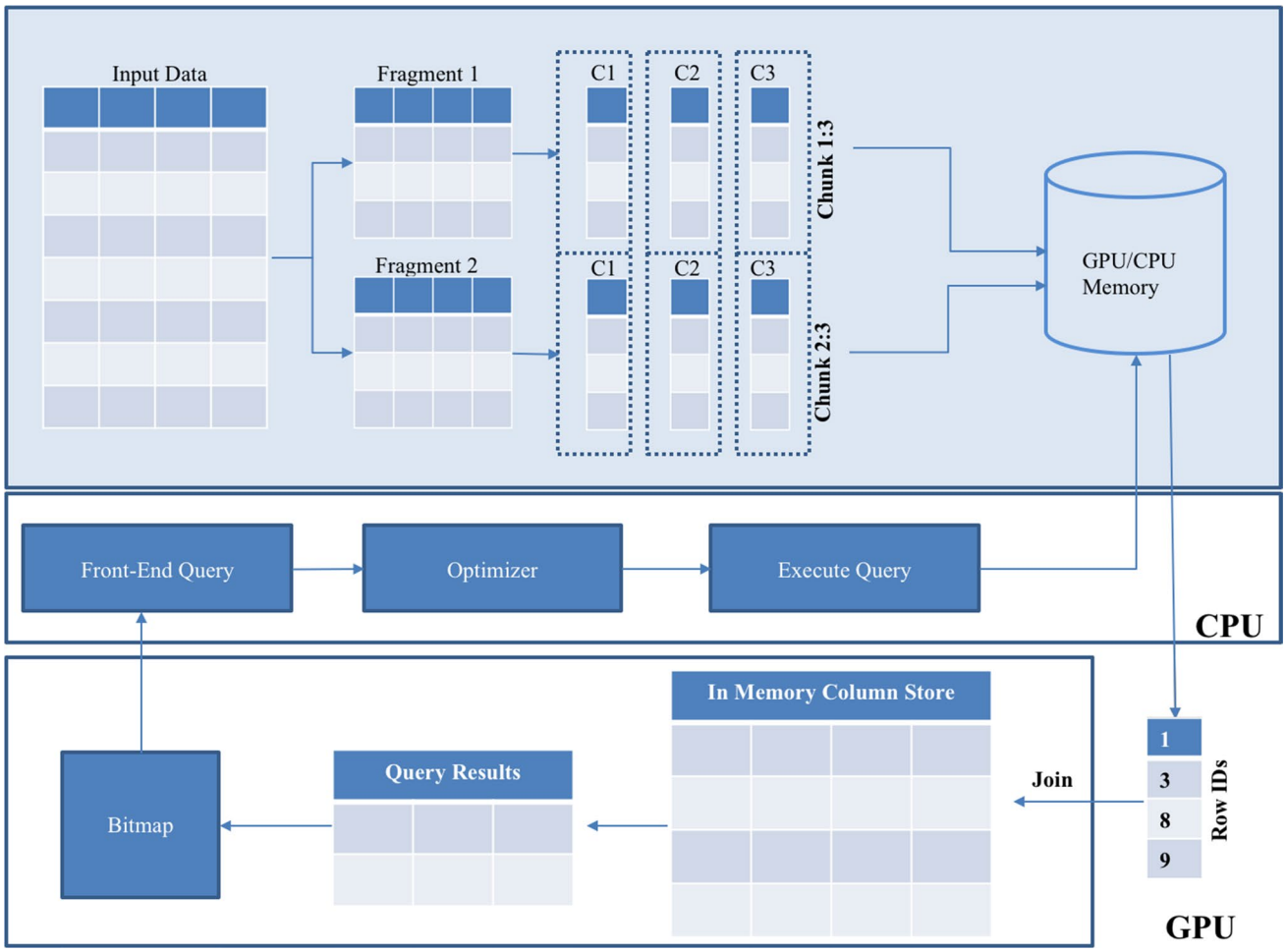


Fig. 1 Design architecture for visual analytics

that are infrequently accessed. CPUs are more efficient at processing text information; hence, as shown in the decision matrix, string column types are usually stored on the CPU.

CPU–GPU Data Processing and Rendering

The key data processing routines carried out on this platform include reductions, aggregations and filtering. Data processing is typically triggered by a user interaction on the front end. Once a query is submitted, a query optimizer determines the right sequence to execute the query, finds the location of queried columns (CPU or GPU), and finally generates and compiles code to run the query. The compiled codes typically run a map-reduce code on multi-node Cassandra CPU cluster and a parallelized SQL code on the GPU. Compiling codes during runtime can drastically slow down response rates for each query. To overcome this bottleneck, for each database created, a code compiler engine generates and pre-compiles both CPU and GPU codes for all possible queries that could be submitted by a user. Hence, at runtime, the query optimizer only needs to find the right codes and where to run them. This design improves query performance significantly.

Processed data can be rendered and manipulated on the front-end module. The visualization framework is able to render millions of data points and produce complex visualizations by leveraging the power of the back-end GPU database architecture. Rendering all charts on the GPU server is, however, not practical, because of memory limitations. As a result, our design uses the browser with a CPU back end to render simple charts such as histograms, bars, lines and pie charts. By using React (React 2013), to juxtapose both complex and simple charts in a single dashboard, and the cross-filter model (Crossfilter 2012), to filter across different charts, we are able to provide multi-dimensional insights into large datasets.

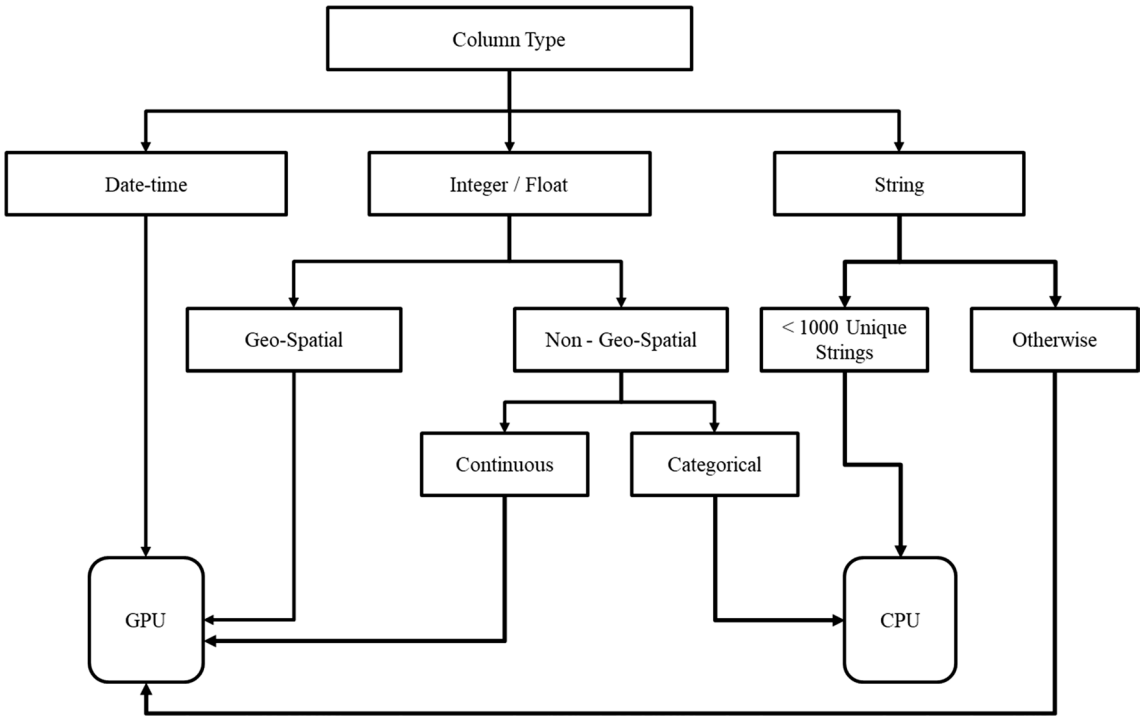


Fig. 2 Decision flow chart for prioritizing which columns are stored on GPU vs CPU

Point and Line Maps

OpenGL is used to render all geospatial datasets. It is able to consume and render millions of points or lines on the GPU server side within a fraction of a second. Rendered results are compressed (to reduce the size of data transferred on the network) and pushed to the front end as a rasterized PNG image. On the front end, Mapbox GL is used to create an interactive model of the PNG image by overlaying it on a base map and adding functions such as zooming and filtering. Because Mapbox GL uses WebGL for image rendering, it is fast and introduces very low latencies in the front end. Figure 3 shows a map rendering of 48 million data points of real-time bus trajectories in the city of St Louis over a 1 month period.

MapGL enables manipulation of the map visualization at the finest scale with different types of filters. This is extremely relevant especially for large data exploration. The platform has three main tools for filtering chart views: circular, polyline and lassor. Figure 3b shows some examples of the different types of map manipulation tools. MapGL is also scale independent; hence, different zoom levels can be used on the fly.

Binned Charts

The current framework is designed to render binned charts for both categorical and continuous data types. For categorical (and ordinal) data types, each distinct value is treated as a bin, whereas for continuous data, data is grouped into adjacent intervals over a continuous range. Depending on the complexity of the visualization, binned charts could be rendered on the GPU or CPU server side. The heatmap shown in Fig. 4 for example displays traffic speed on an interstate highway at 1-mile intervals over a 1 year period. For a 270 mile stretch of road, this generates over a million points even after binning. Rendering inside the browser will negatively impact the ability to provide real-time interactivity. GPU server-side rendering is therefore a perfect fit for this case. For simple charts requiring minimal data as shown in Fig. 4, D3 (Bostock et al. 2011), a Javascript library which uses HTML, SVG and CSS for rendering charts is used.

Temporal and One-Dimensional Charts

Similar to simple binned charts, temporal and one-dimensional values are rendered in the browser using D3. Example line charts shown in Fig. 4 are typically used to visualize temporal datasets. We designed them to have brush handles which can be used in active views to narrow analysis within a particular range. Temporal values can also be binned at different levels of granularity: yearly, monthly, daily or hourly.

Finally, we utilize React to build UI components that uses the cross-filter model to apply filters across all the different charts in the dashboard. This allows for seamless and intuitive analysis of multidimensional datasets. The following

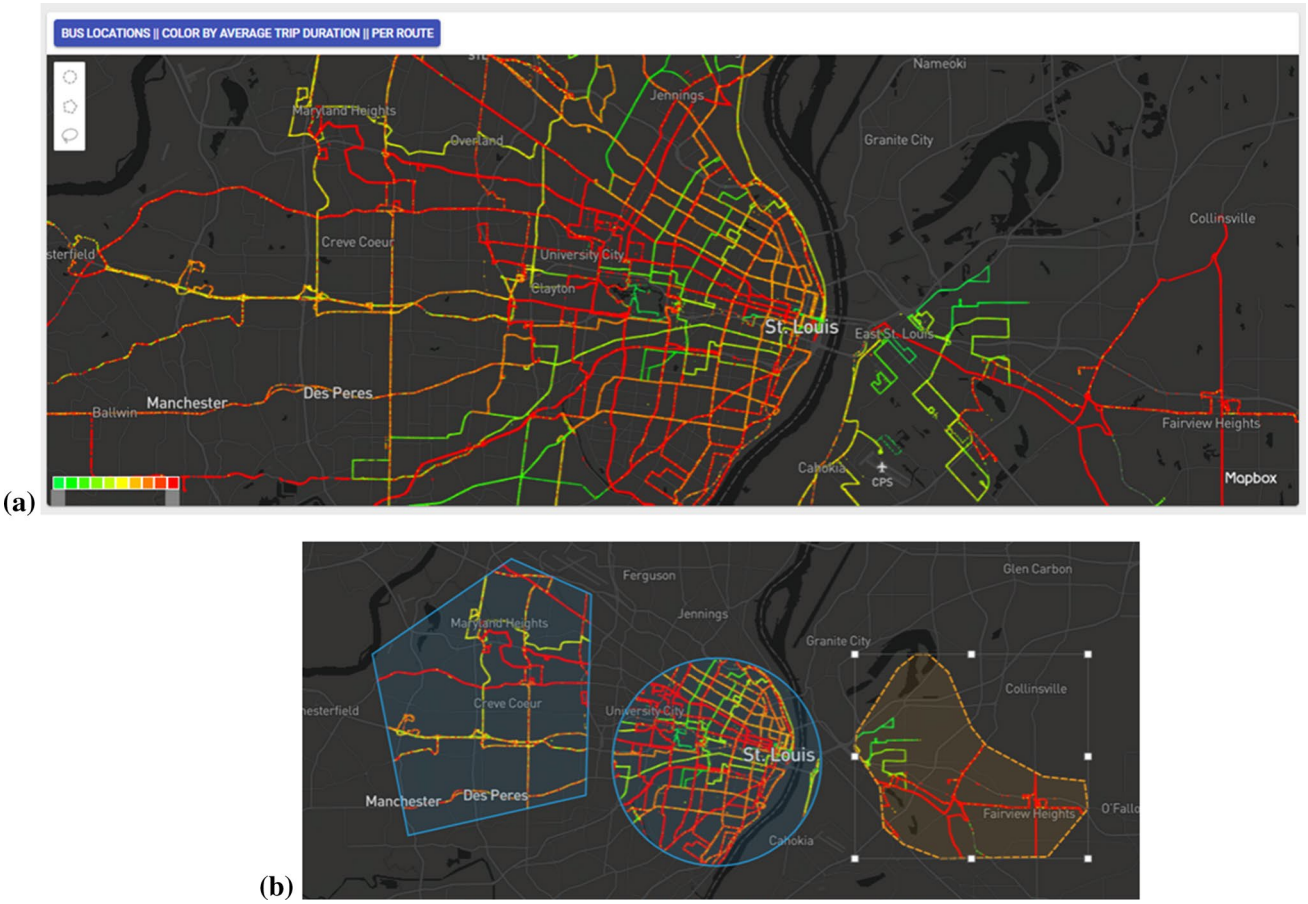


Fig. 3 Map rendering of bus locations in St Louis, Missouri

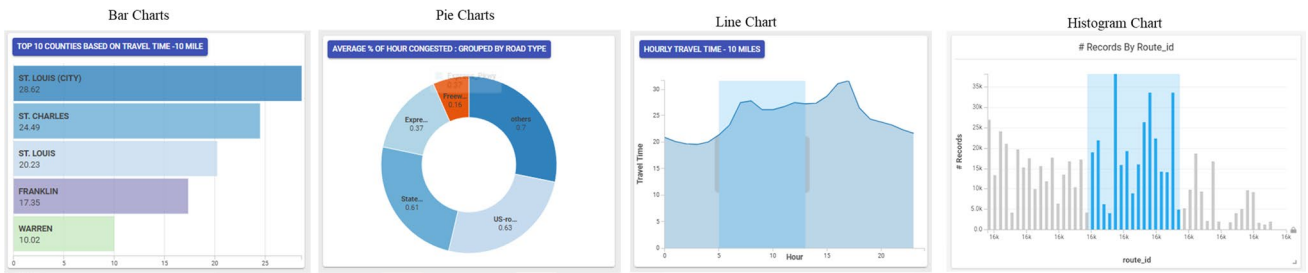


Fig. 4 Examples of binned and temporal one-dimensional charts created with D3

section shows examples of interactive visualizations tools created with different transportation big data applications.

Transportation Visualization Examples

In this section, the visualization framework developed is used to create applications for traffic mobility–safety operations and transit performance monitoring. We selected these two areas of transportation because the volumes of data generated by transit and traffic operations are so huge that conventional, off-the-shelf visualization tools are unable to provide fine-scale analysis of this data. These reasons make these datasets perfect examples for evaluating the effectiveness of our developed framework. The attributes of the data used to create the applications are shown in Table 1. The traffic data reports traffic speed and travel time information for each segment of road in the state of Missouri. The data is collected through a probe technology which acquires traffic-related data from GPS-enabled devices such as vehicles

Table 1 .

	Traffic data	Transit data	Crash data	Weather data
Duration	4 years	3 months	4 years	4 years
Data resolution	60 s	30 s	Daily	Daily
Data coverage	Missouri state	St. Louis	Missouri state	Missouri state
Data size	140 GB	65 GB	18 GB	200 MB
# of columns	16	18	38	12
# of rows	186 million	38 million	1.7 million	15,000

and cell phones. The transit data is obtained through the General Transit Feed Specification (GTFS). It captures real-time locations of busses and other attributes such as delays, stops and routes. Archived crash and weather datasets were ingested from transportation management system feeds into a GPU–SQL and Cassandra database. The GPU database writes data about 25,000 rows per second, while the Cassandra database writes at 15,500 rows per second. Real-time data ingestion is currently not supported by the framework.

Traffic Mobility–Safety Operations

This impact of road crashes on mobility or vice versa is very important for estimating the cost of a crash or the benefits of mobility improvements. To perform such analysis, the mobility data (probe data) should first be integrated with the crash data. A spatial conflation model was developed to carry out this integration process. The result is a mapping between probe segments and accident locations. A detailed explanation of the conflation model is beyond this paper. The integration of both datasets resulted in a unified data with 246 million rows which was consumed by the framework for visualization. A snapshot of the interactive dashboard for exploring crash and mobility data is shown in Fig. 5. The basemap is filtered to show all crashes that occurred on a particular route (IS70). The heatmap shows the impact of the crashes on mobility along the selected route over time. The remaining row charts display statistics on the type of crashes and the road weather conditions.

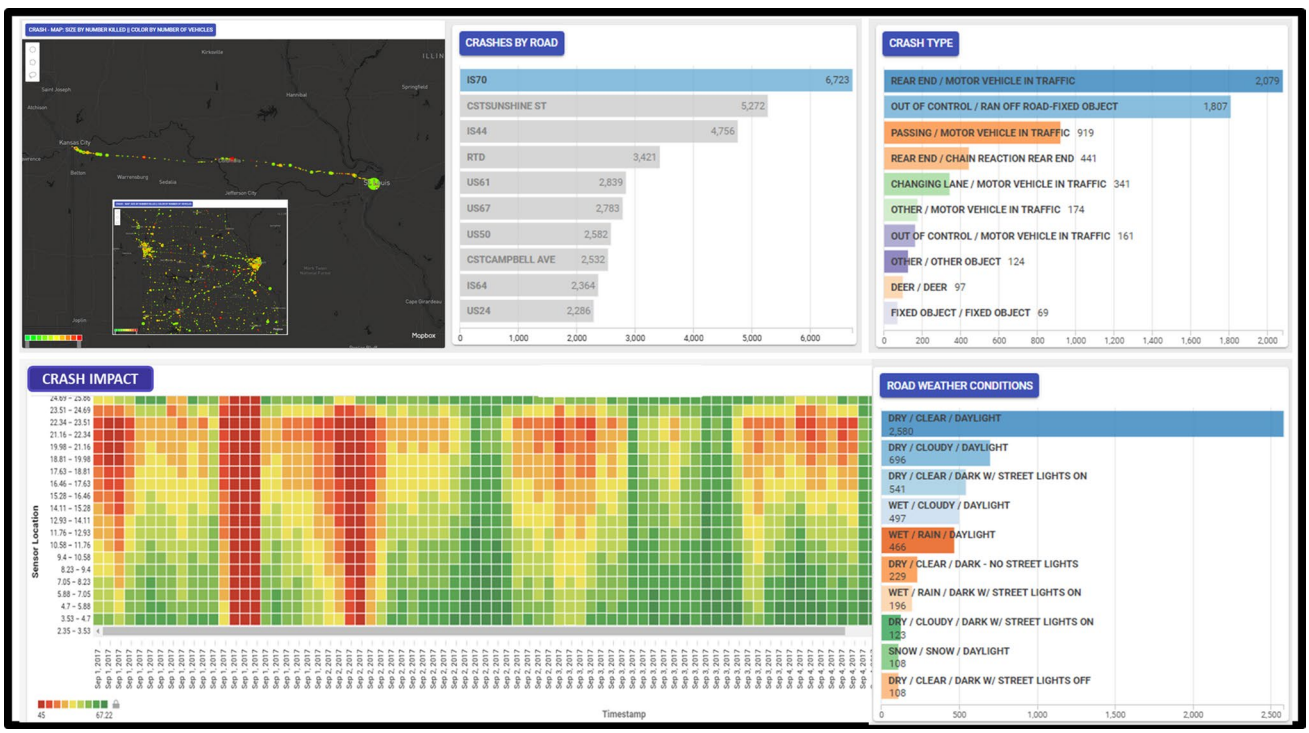


Fig. 5 Visual analytic dashboard for traffic mobility and safety

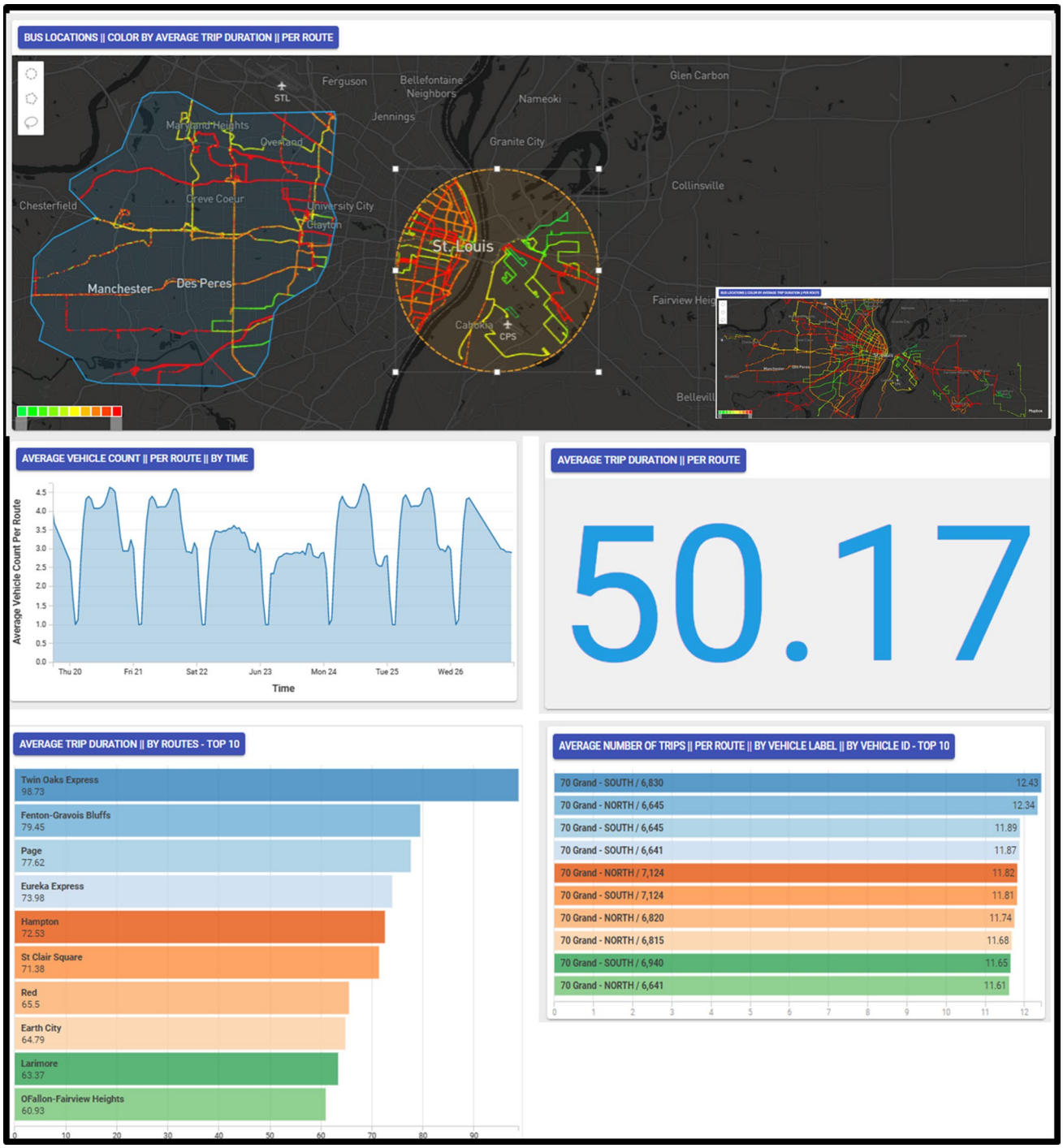


Fig. 6 Visual analytic dashboard for traffic performance assessment

Transit Performance Assessment

The transit visual analytics dashboard shown in Fig. 6 is designed for assessing the performance of transit systems such as bus lines, or evaluating accessibility issues related to transit. The transit application also required integration of both transit and mobility data. This enables the system to compute reliability of bus routes based on traffic conditions. The duration of data collected for this application is 3 months. The integrated data had approximately 98 million rows. The map shows the trajectory of each bus line, colored by the reliability of the route which is a function of actual bus delays and the variance of route travel time. A circular and lasso filter is used to select regions of interest from the

map chart. The time chart is zoomed in to capture daily transit patterns for the filtered regions of interest.

A discussion on the speed at which the developed framework enables interactivity, data size requirements and hardware architectural designs follows in the next section.

Performance Evaluation

A series of experiments were conducted to develop benchmarks and evaluate the performance of the visual analytics platform, and also develop benchmarks for comparative analysis with other legacy visualization frameworks. In each case study, the observed latency (measured as the time taken to compute and render a display) is used as the key performance measure. The datasets used in all experiments are a subset of the data shown Table 1.

Computing Environment

Our experiments were run on a three-node cluster, each equipped with Intel core i7 processor, 500 GB of SSD storage and NVIDIA GeoForce GTX 1080Ti graphics cards with 11 GB memory. For comparative analysis, a single node, bare metal machine with eight cores, 64 GB of memory, 1 TB of SSD storage and an NVIDIA GTX 1080Ti graphics card was used.

Data Size Effect

Our first experiment evaluates the influence of a number of rows in a table on the analytic speed of the framework. Figure 7a shows the latencies observed as we varied the number of rows in the traffic data (see Table 1) from 5 to 100 million rows. In this experiment, we limited the number of charts to only two: a map chart of road segment locations and a row chart of road type. For each filter applied on the row chart, we recorded the time taken to compute and render the map display. From the figure, it can be observed that the framework takes less than 0.1 s to respond to queries on a table consisting of at most 100 million rows. The relationship between compute and render speed as the size of data increases is worth noting: for medium- to large-sized tables, a significant proportion of the latencies are due to compute, whereas for small-sized tables, the time taken to render a display takes almost double the time to compute. Compared to compute time, data rendering time tends to be more stable even with increasing data size.

Chart complexity

Although we achieved very quick response rates in the preceding experiment, the benchmarks were obtained using only two charts. The number and complexity of the charts used in a dashboard could influence these rates significantly. In this experiment, we create a dashboard with six of the most complex charts: one map, one heatmap, two line charts, one scatter plot and one histogram plot. We programmatically brush the line charts and compute the time taken to update the remaining four charts. Figure 7b shows the average compute and render times for different data sizes. Although the response rates are still appreciable, it can be observed that the use of complex charts can increase the response time by as much as 100× on the same size of data. Heatmaps tend to have the highest latencies; lagging behind geographic maps by about 3 s. Increasing the bin sizes for heatmaps could improve the response rates.

Query Complexity

The influence of the complexity of a query is another factor that was evaluated. We divide queries into three levels of complexity: level 1 query is when a single filter is applied to a single chart, level 2 applies two or more filters to simple charts (row charts, histograms, table), and level 3 applies two or more queries to a mixture of simple and

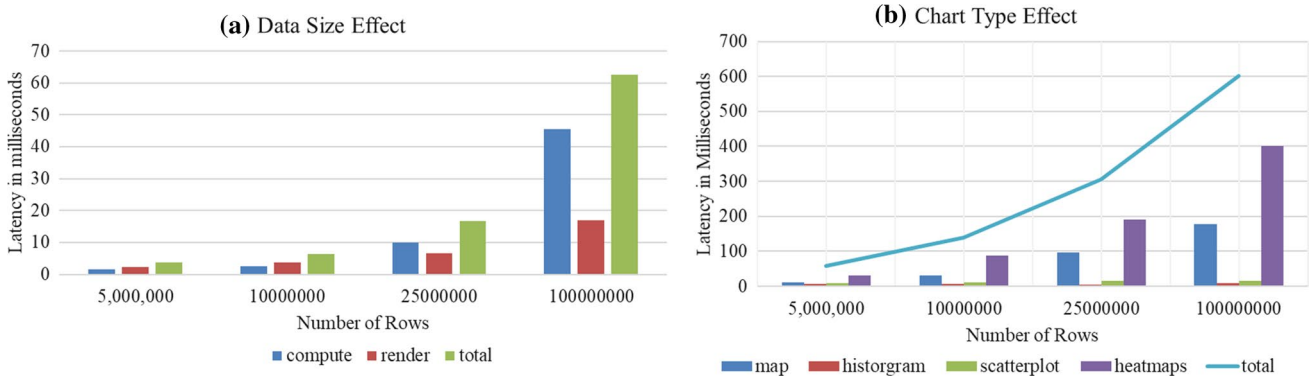


Fig. 7 Influence of data size and chart complexity on query response rates

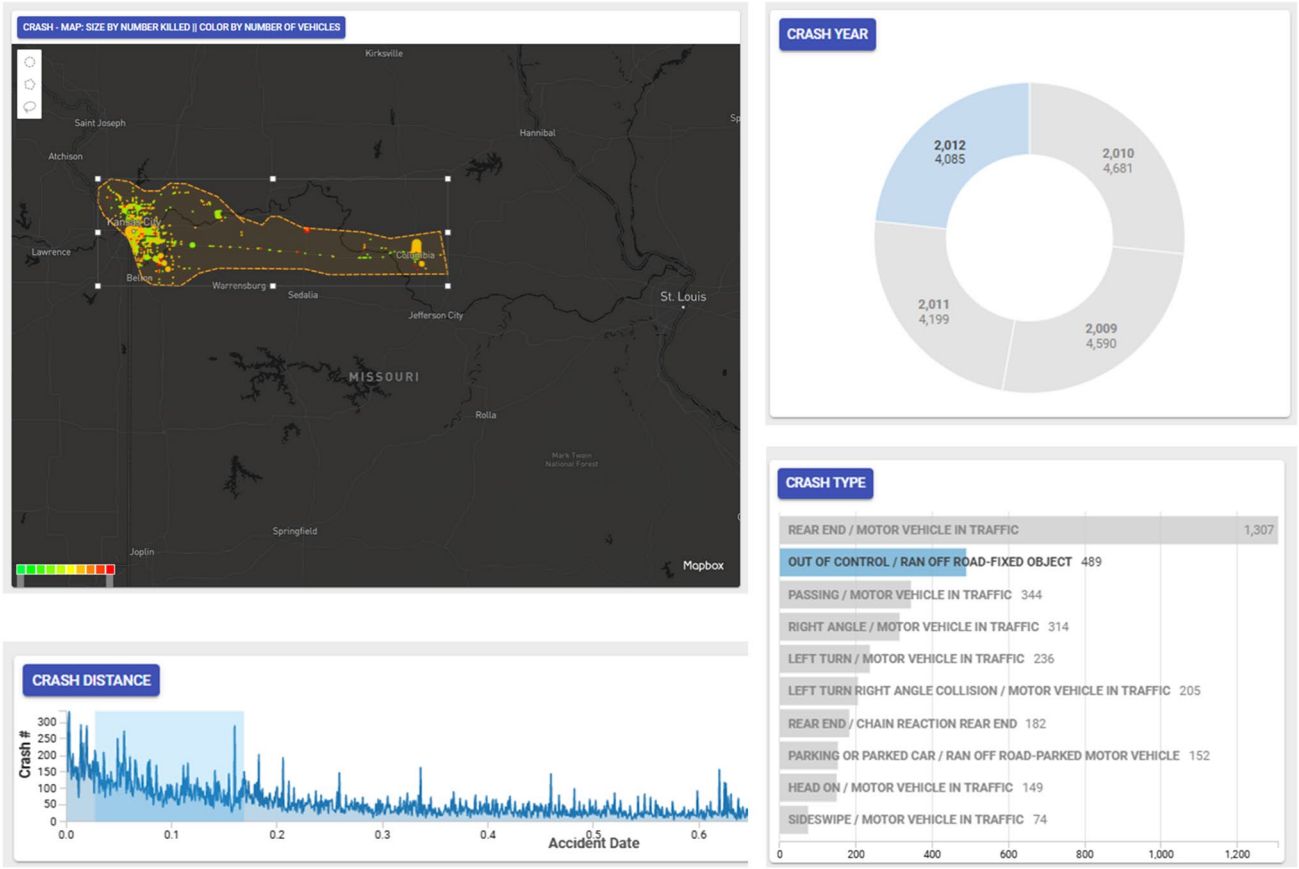
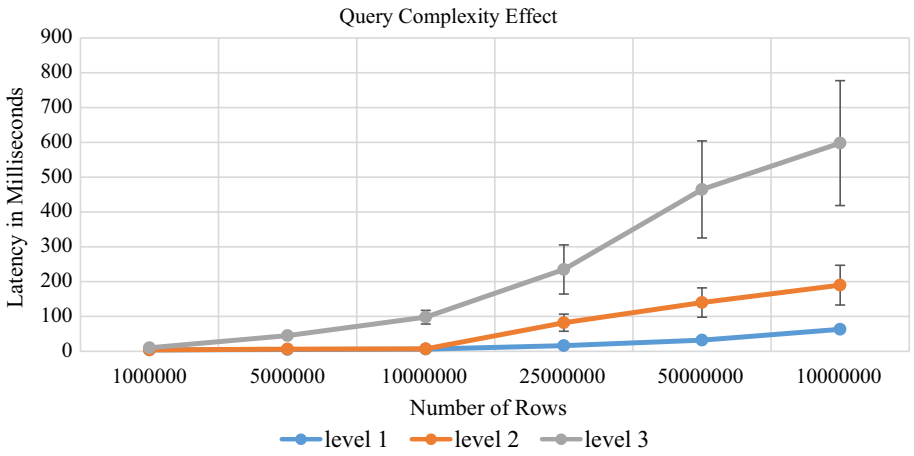


Fig. 8 Query complexities: example of a level 3 query type

Fig. 9 Effect of query complexity of response rate



complex charts. Figure 8 shows an example of a level 3 query. Filters are applied to the map, line, row and pie charts one at a time. Figure 9 shows average response rates for levels 2 and 3 type queries on the traffic, transit and crash datasets shown in Table 1.

Comparative Analysis

In this final section, we compare the visual analytics framework developed with two legacy CPU-based systems

Table 2 Comparative analysis of developed framework with tableau and D3

Query complexity	Chart types used	Number of rows	D3 + Cross-filter + MongoDB	Tableau	CPU–GPU framework
Level 1	Map, row and line charts	< 50,000	–	–	–
		100,000	–	–	–
		500,000	1.2 s.	1.3 s.	–
Level 2	Map, row, line, heatmap, and scatter plot	< 500,000	5.6 s.	5 s.	–
		1000,000	12.15 s.	10 s.	–
		5,000,000	15.34 s.	13 s.	1.06 s.

used by transportation agencies: Tableau and D3 with cross-filter. We re-created the visualization in Fig. 5 on both frameworks and applied a mixture of level 1 and 2 queries to evaluate the response rates. It should be noted that the data size for this experiment was reduced to 5 million rows due to the limitations of CPU memory. Also, the proposed framework is implemented on a single GPU server instead of a cluster. For D3, we had to support it with MongoDB back end to process more than 1 million rows of data. Table 2 shows query details and the resulting display latencies. Note, since there was no programmatic way to calculate the latencies for Tableau, we report only instances where the response rates were more than 1 s. From the table, it is evident that leveraging a CPU–GPU architecture significantly improves the level of interactivity for large datasets. Tableau is slightly faster than D3 especially for complex queries.

The framework developed is also compared with a GPU-based visual query platform called imMens, developed by Liu et al. (2013). It is arguably one of the first frameworks developed to enable real-time visual querying on large datasets. Because their implementation leverages the GPU only for data rendering and reductions, it overcomes the latencies due to data transfers between CPU and GPU. It ability to integrate multivariate data tiles and parallel processing significantly improves interactivity speed over large datasets. To compare the two frameworks, we measure the latency in chart rendering by the number of frames rendered per second (frame rate), instead of the response time as used in preceding experiments. This is because, in general, the differences in response time for GPU queries tend to be very small, and using the frame rate enables us to quantify the small differences in performance between the two frameworks.

We varied the size of data from 1 to 100 million rows. Other elements such as chart types, number of charts and query levels (1–3) were also varied. The results of our comparison are shown in Fig. 10. From the figure, when the number of rows is less than 5 million, imMens tends to be superior irrespective of the complexity of the query, the number of charts or the chart type used. In fact, at 5 million or less rows of data, imMens can render data at 10 fps faster

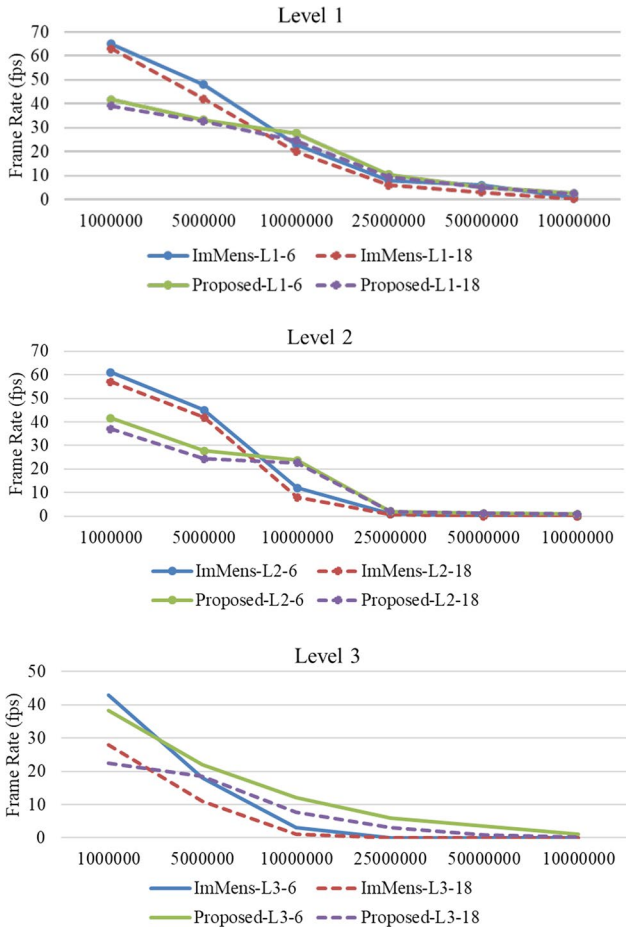


Fig. 10 Comparison of the proposed visual analytic framework with imMens

than the proposed framework. The response rate for the proposed framework, however, exceeds 25 fps when the number of rows of data is less than 10 million. Therefore, the gains by imMens do not present any significant visual differences during querying. Beyond 25 million rows of data, as the complexity of queries and number of charts used increases, the performance of imMens drops significantly compared to the proposed framework. This trend might be due to the fact that GPU memory for imMens is used up, while the

proposed framework takes advantage of the CPU Cassandra cluster to support the GPU. It is also important to note that the number of charts used in the visualization has marginal influence on the imMens compared to the proposed framework. For level 1 type queries on data with 100 million rows, imMens is able to respond to complex queries at 0.1 fps compared to 2 fps by the proposed framework.

Concluding Remarks

The current paper outlines the development of large data visual analytics framework for transportation systems. It leveraged the parallel processing power of GPUs and the high memory bandwidth of commodity CPU clusters to visualize, interact and analyze big transportation datasets in the browser at rates 100× faster than legacy CPU platforms. The framework developed first ingests large tables as column chunks into a hybrid CPU–GPU database architecture. A decision matrix is used to prioritize which columns are stored in GPU or CPU memory. Highly parallelized codes are used to process data simultaneously on CPU and GPU after queries are initialized. The processing results are compressed into image files and transferred to the front end for visualization. The browser and back-end CPUs are used to handle rendering of simple charts such as histogram, line and row charts, whereas the GPU memory is dedicated to rendering complex charts such as maps, heatmaps and scatter plots.

A series of experiments were conducted to evaluate the effect of data size, chart and query complexity on the response rate or latency of the developed framework. For tables with at most 100 million rows of data, we are able to achieve query response rates less than 100 ms. As chart and query complexities increase, GPU memory is overtasked, leading to significant latencies in computations and rendering on the front end. In some cases, observed latencies increased to about 6000 ms for dashboards with multiple heatmaps, geospatial maps and at most 100 million rows of data. Our final experiments compared the methodology with two conventional CPU platforms: D3 and Tableau. On relatively smaller-sized datasets, the query response rates for the developed framework was about 10× faster than both CPU platforms.

A key limitation of the current framework is its inability to handle non-structured data. It assumes that the data exists in a tabular form and does not contain data types such as video and images. Future studies will incorporate pipelines for handling non-structured data. Also, the current visual analytic framework is designed for analyzing batch-historical data. An extension for ingesting real-time ingestion and visualization of data simultaneously will be investigated.

References

Abdullah K, Fabio M, Kaan O, Claudio TS (2017) Data visualization tool for monitoring transit operation and performance. In: 5th IEEE international conference on models and technologies for intelligent transportation systems (MT-ITS)

Adu-Gyamfi YO, Sharma A, Knickerbocker S, Hawkins NR, Jackson M (2016) A comprehensive data driven evaluation of wide area probe data: opportunities and challenges. In: Civil, construction and environmental engineering conference presentations and proceedings. 38. https://lib.dr.iastate.edu/ccee_conf/38

Andrienko G, Andrienko N, Chen W, Maciejewski R, Zhao Y (2017) Visual analytics for transportation: state of the art and further research directions. IEEE Trans Intell Transp Syst. 18(8)

Badu-Marfo G, Farooq B, Patterson Z (2019) A perspective on the challenges and opportunities for privacy-aware big transportation data. J Big Data Anal Transp 1(1):1–23

Bostock M, Ogievetsky V, Heer J (2011) D3 data-driven documents. IEEE Trans Visual Comput Graph 17(12):2301–2309

Brennan TM, Gurriell RA, Bechtel AJ, Venigalla MM (2019) Visualizing and evaluating interdependent regional traffic congestion and system resiliency, a case study using big data from probe vehicles. J Big Data Anal Transp 1(1):25–36

Bureau of Transportation Statistics (BTS) (2019) Overview of US—North American Freight by Port, State and Mode. https://explore.dot.gov/t/BTS/views/Dashboard_StatebyPort/Overview?%3Aiid=3&%3AisGuestRedirectFromVizportal=y&%3Aembed=y&%3AusingOldHashUrl=true. Accessed July 2019

Chen W, Guo F, Wang F (2015a) A survey of traffic data visualization. IEEE Trans Intell Transp Syst. 16(6)

Chen L, Chowdhury A, Loulakis C, Ownes M, Thorisson H, Connelly E, Tucker C, Lambert J (2015b) Visualization of large data sets for project planning and prioritization on transportation corridors. IEEE Systems and Information Engineering Design Symposium, Charlottesville

Crossfilter (2012) Fast multidimensional filtering for coordinated views. https://dc-js.github.io/dc.js/

Heer J, Shneiderman B (2012) Interactive dynamics for visual analysis. Queue 10(2):30

IowaDOT (2018) Realtime analytics of transportation data. https://reactor.ctre.iastate.edu/iwz-crash/. Accessed July 2019

Islam J, Sharma A (2019) A cyber infrastructure for big data transportation engineering. J Big Data Anal Transp 1(1):83–94

Liu Z, Jiang B, Heer J (2013) imMens: real-time visual querying of big data. Comput Graph Forum. https://doi.org/10.1111/cgf.12129

Mi P, Sun M, Masiane M, Cao Y, North C (2016) AVIST: a GPU-centric design for visual exploration of large multidimensional datasets. Informatics. https://doi.org/10.3390/informatics3040018

Moritz D, Howe B, Heer J (2019) Falcon: balancing interactive latency and resolution sensitivity for scalable linked visualizations. In: Proceedings of the 2019 CHI conference on human factors in computing systems, paper no. 694. ACM, NY, USA

Mostak T (2014) An overview of MapD (Massively Parallel Database). http://www.smallake.kr/wp-content/uploads/2014/09/mapd_overview.pdf. Accessed July 2019

Mostak T (2016) Using GPUs to accelerate data discovery and visual analytics. In: Future technologies conference, San Francisco, US, December 2016

Nancy M (2018) Why visual analytics, Tableau White Paper. https://cdn2.hubspot.net/hubfs/2383378/Tableau%20Whitepaper%20-%20Why%20Visual%20Analytics.pdf?t=1520904633993. Accessed July 2019

NHTSA (2016) Traffic fatalities in crashes involving speed. https://icsw.nhtsa.gov/nhtsa/fars/speeding_data_visualization/. Accessed July 2019

Picozzi M, Verdezoto N, Pouke M, Vatjus-Anttila J, Quigley A (2013) Traffic visualization applying information visualization techniques to enhance traffic planning. In: international conference on computer graphics theory and applications and international conference on information visualization theory and applications. Barcelona, Spain. pp 554–557

React (2013) A javascript library for building user interfaces. https://reactjs.org/. Accessed July 2019

Sharma A, Ahsani V, Rawat S (2017) Evaluation of opportunities and challenges of using INRIX data for real-time performance monitoring and historical trend assessment. Reports and White Papers. 24. https://lib.dr.iastate.edu/ccee_reports/24 Transportation Systems, Journal of Sensors, 19(332)

Sobral T, Galvão T, Borges J (2019) Visualization of urban mobility data from intelligent sensitivity for scalable linked visualizations. In: CHI conference on human factors in computing systems proceedings, Glasgow, Scotland, UK. Source available at: https://square.github.io/crossfilter/. Accessed July 2019

Valerie L, Denis G (2014) Visual analytics for cyber security and intelligence. J Def Model Simul 11(2):175–199

VDOT (2015) Crash analysis tools. https://public.tableau.com/profile/tien.simmons#!/vizhome/Crashtools8_2/Main. Accessed July 2019