Prototyping A Web-based High-Performance Visual Analytics Platform for Origin-Destination Data: A Case study of NYC Taxi Trip Records

Jianting Zhang
Dept. of Computer Science
City College of New York
New York City, NY, 10031
jzhang@cs.ccny.cuny.edu

Simin You
Dept. of Computer Science
CUNY Graduate Center
New York, NY, 10016
syou@gc.cuny.edu

Yinglong Xia
IBM T.J. Watson Research Center
Yorktown Heights, NY, 10598
yxia@ us.ibm.com

ABSTRACT

Origin-Destination (OD) data has quickly emerged as a popular and fast growing spatiotemporal data type due to widely adoption of GPS, smartphones and location dependent social networks. Several previous works have developed techniques for managing and visualizing large-scale OD data in desktop computing environments. In this study, by leveraging our experiences in Web-GIS and parallel spatial data processing and learning from successful OD data visualization case studies, we have developed a Web-based high-performance visual analytics prototype platform for OD data. Observing that interactive spatial queries typically only involve a limited number of Regions of Interests (ROIs), we propose a simple yet effective technique to aggregate OD records into dynamically defined OD polygons by data parallel scanning OD point locations for cache efficiency and easy parallelization on conventional multi-core hardware for high efficiency and performance. By dynamically integrating with a graph database backend, our prototype platform is capable of visualizing social network analytical results and guide users to further retrieve detailed information of interests. Two experiments are provided to demonstrate the utilization of the proposed techniques, including web frontend functionality and backend efficiency, by using more than 170 million taxi trip records in NYC in 2013 as well several urban infrastructure datasets. Interactive demonstrations are available for the web-based system.

1. INTRODUCTION

Approximately 13,000 taxicabs in the New York City (NYC) equipped with GPS devices generate more than half a million taxi trip records per day with each trip has a pickup and a drop-off location[1]. As of 10/07/2014, there are more phones (7.2 billion) than people on the Earth[2]. Cell phone call logs with caller-callee locations represent a category of data at an even larger scale. Also as visitors travel around the world more frequently, the increasingly popular location-dependent social networks such as Foursquare, location-enhanced social media such as text posted to Wiki sites and images and videos posted to Flickr and YouTube, record more and more location data at different granularities and accuracies. All the above data have one characteristic in common, i.e., recoding Origin (O) and Destination (D) locations, and they are typically termed as OD data.

The inefficiency of using traditional GIS, Spatial Databases or Moving Object Databases to manage large-scale OD data has motivated us to develop new parallel techniques on modern hardware, including multi-core CPUs and Graphics Processing Units (GPU [1]) and distributed computing systems to process the OD data efficiently on modern hardware [2] [3]. While experiments have demonstrated high-efficiency, previous works mostly focus on spatially joining point location data with urban infrastructure data (or spatial join [4]) in an offline manner and do not support online visual analytics for interactive query and analysis. On the other hand, as visual analytics typically has a local focus (e.g., Region of Interests - ROIs), the computing demands is generally less intensive than globally exhaustive searching over a large OD data repository. Despite it is still technically challenging to make full use of commodity parallel hardware for high performance, it is possible to utilize conventional hardware to support interactive query processing and analysis based on limited ROIs. Different from offline spatial join processing, visual analytics of urban OD data involves considerably more operations, such as various social network algorithms in computing centralities and ranking, to guide effective visual exploration. Although these algorithms are well-defined and open source implementations are available, the integration of spatial and spatiotemporal query processing and social network analytics with a visual exploration interface is non-trivial from a system development perspective. Furthermore, while desktop computing based information visualization system can generally provide more functionality (e.g., TaxiVis [5]), a web-based platform that allows users to access anytime and anywhere is more desirable, especially for non-expert users.

Towards this end, we have started the initiative of developing a web-based high-performance visual analytics platform for urban OD data. The platform leverages our experiences in processing large-scale OD data on parallel hardware [2]. It integrates Google Map API[3] for map-based visualization at the frontend and IBM SystemG API[4] for social network analysis on graphs at the backend. The platform supports ROI-based visual queries on Google Map not only for arbitrarily drawn polygons but also for OD polygon pairs, with a response speed in a fraction of a second in a web based computing environment. The platform populates offline generated geosocial graphs into IBM SystemG graph databases and provides social network analytical services through its web-enabled APIs. The social network analytical results are subsequently geocoded and visualized in Google Map based web frontend for subsequent map-based interactive visual exploration. The communications between the Google Map based frontend and the respective distributed backend services are through the open standard JSON format to lower system development barriers and achieve interoperability, scalability and high performance.

While our technical contributions in this study are mostly on the system designs and implementations, our prototype platform is unique in several aspects: an in-house developed spatial query processing backend that utilizes in-memory and parallel techniques for performance, integrating with industrial strength social network analysis modules for enhanced functionality that is typically not available in a web-based GIS, and, a visual analytics frontend specially designed for OD data on top of Google Map APIs that is easy to use. To the best of our knowledge, we are not aware of previous works that have explored similar system designs for large scale urban OD data. The rest of the paper is arranged as follows: Section 2 introduces background and motivation and briefly discusses related work. Section 3 presents system designs and implementation details. Section 4 provides case studies of several application scenarios to demonstrate the functionality of the prototype platform. Finally Section 5 is the conclusion and future work directions.

2. BACKGROUND, MOTIVATION AND RELATED WORK

The work being reported is related to quite a few disciplines and their intersections, including GIS, Big Data, visual analytics, social networks and web technologies. Web-GIS has established itself as a mature application domain to publish georeferenced data over the Web in the past two decades. In addition to open source software stacks, such as PostgresSQL/PostGIS [6] - MapServer[5]/GeoServer[6] - OpenLayers[7], major GIS vendors and their products, such as ESRI ArcGIS Server[8] and its client SDKs, allow users to publish their data as web services and consume such services in web-based environments for visualization, analytics and their combinations. Typically the backend has a database or data repository module to manage vector and raster geospatial data, and a middleware module to transform the raw geospatial data to various open formats, including images and documents using markup languages, such as Geographical Markup Language (GML[9]) and Keyhole Markup Language (KML[10]). The web frontend is typically provided as Javascript libraries or other web-plugin libraries (such as Microsoft Silverlight, Oracle JavaFX and Adobe/Apache Flex) that can assemble various formats of georeferenced data transmitted over the HTTP protocol and graphically represented them in browsers. Standardization organizations, such Open Geospatial Consortium (OGC[11]), have been playing an import role in standardizing the interfaces among different components in a Web-GIS application, such as Web Map Services (WMS[12]) and Web Feature Services (WFS[13]), to achieve interoperability among modules provided by different software vendors. The popular Google Map and Microsoft Bing Map combine proprietary data and a limited subset of GIS functionality as web services and provide such services through their frontend Web APIs. While the commercial products make it easy to use services they provide, e.g., using web maps as background for visualization purposes, typically it is difficult to publish users' own data to use similar services or to significantly extend the services for more complex analysis. On the other hand, although most of existing open source or commercial Web-GIS software allows users to store large-scale georeferenced data in disk-resident databases or data repositories, their performance is not acceptable for large-scale data [2] due to the poor scalability of software of multiple layers in Web-GIS software stacks. Furthermore, while Google Map and Microsoft Bing Map may be able to optimize their data accesses and services for the data they directly manage, as users are required to integrate their customer data and services at the frontend (e.g., based on Javascript), the performance degenerates quickly as customer data and services increase. As a consequence, existing web-based systems for OD data visual analytics (e.g., [7]) have not utilized Web-GIS technologies extensively beyond simple web map overlay for visualization. Indeed, without significantly changing backend and/or frontend in the existing Web-GIS software stacks, it is very difficult to handle large-scale OD data in a sensible way. As a preliminary remedy, our prototype platform reuses Web-GIS frontend (Google Map API in particular) but enhances the backend with a high-performance in-memory parallel data management system to perform spatial queries, which will be further discussed next.

Information visualization and visual analytics techniques specially designed for spatiotemporal data, trajectory data and OD data, which are becoming increasingly popular in the past few years, can be considered as extensions and enhancements to traditional GIS. While traditional GIS nicely integrates data management, visualization, analysis and simulation functionality into software suits and provide end-to-end solutions for many real world applications in the past few decades, they have not provided sufficient functionality for spatiotemporal data in general and OD data in particular. There have been a plethora of works on managing, analyzing and visualizing spatiotemporal data and trajectory data and we refer to [8] for reviews and examples. The seminal work of TaxiVis reported in [5] has motivated quite a few works to develop visualization gadgets specifically for OD data. As an example, a recent work in [7] has developed Circular Pixel Graph and Spatio-Temporal Stacked Graph to better understand patterns from OD data. Similarly, OD-Wheel [9] was designed to explore the temporal dynamics of OD clusters to help users to study traffic pattern of a ROI. Different from TaxiVis that allows user to dynamically define ROIs by interactively drawing polygons on a base map and retrieve taxi trip records from the backend database on-demand, which involves considerable development efforts in indexing and query processing, the works reported in [7] and [9] seem to rely on preprocessing to aggregate OD data to a limited number of predefined regions to reduce computing demands. While effective from a computing perspective, the techniques might limit their capabilities to support fine-grained spatiotemporal visual explorations of large-scale OD data due to the predefined aggregations. We note that, all these works are based on a desktop computing environment. While preprocessing techniques, including indexing and aggregation, may utilize distributed computing environment, query processing and visualization in these studies seem to be based on a serial computing model, which may limit their scalability and achievable performance. In contrast, our prototype platform is designed to be web-based and naturally supports distributed computing environments. In addition to dispatching multiple queries to distributed backend servers, each server can natively utilizes multi-core CPUs for efficient parallel query processing.

Large-scale OD data such as taxi trips has also motivated several research works from spatiotemporal data management perspective, which is closely related to data management in GIS. We have developed several data parallel spatial indexing and query processing techniques both on GPUs and GPU-accelerated clusters and we refer to [3] for a brief summary. Experiments have shown that spatially joining hundreds of millions of GPS OD locations in the NYC taxi trip records with hundreds of thousands of road segments and different types of zones can be completed in the order of tens of seconds. Compared with traditional disk resident spatial databases techniques running a single CPU core, 3-4 orders of magnitude of speedups (from tens of hours to tens of seconds) have been achieved due to the combined improvements of columnar layout, data parallel geometry operation, in-memory processing and many-core GPU acceleration. Based on the results, it is reasonable to assume that, interactive queries that typically involve only a limited number of manually drawn simple ROI polygons, can perform much faster than spatial joins that involve hundreds of thousands of complex real world polygons. This further motivates us to develop a simplified spatial query technique for such interactive queries in the context of web-based visual analytics for OD data. As detailed in Section 3, our point-in-polygon test based query processing technique at the backend does not require sophisticated spatial indexing, such as KD-Tree in TaxiVis [5] and R-Tree, Quadtree or Grid File in our previous works [10] [2]. Instead, our simplified technique simply scans point locations and aggregate OD records that spatially intersect with the MBRs of query polygons by using inexpensive MBR test as a filtering step before performing more expensive point-in-polygon test on point locations and query polygons. The simplified technique is easy to implement and deploy on conventional multi-core CPUs and does not require a GPU. Experiments have shown the response time is typically a fraction of a second for arbitrarily drawn ROI query polygons against the complete 2013 NYC taxi trip dataset whose number of records is over 170 million on a legacy dual quad-core CPU machine. While we acknowledge the need of a comprehensive and powerful backend for large-scale OD data, we believe the simplified spatial query technique is useful as a lightweight module for visual analytics.

While OD data naturally has a spatiotemporal component and can be georeferenced, for a single origin location, the destination location can be arbitrary in a study area. Although road network data can be generally represented as a planar graph where edges only intersect at their endpoints, OD pairs can only be represented as non-planar graphs. In fact, OD pairs that are geographically faraway may have stronger connection. For example, there are much more taxi trips between the Empire State Building (as a ROI) and the JFK airport in NYC. The unique characteristic of OD graphs makes them resemble more to social network graphs. It is thus interesting to apply many well established social network algorithms to better understand OD data. In our previous work, we have aligned the pickup and drop-off locations of NYC taxi trips to road network intersections [2]. We then calculate shortest paths between unique node pairs before aggregating the shortest paths to calculate the centralities of road network segments by empirically assuming the drivers will generally follow shortest paths [11] [12]. A similar idea has been applied to TrajGraph [13] where road segments are partitioned into zones to limit the numbers of OD pairs to ease graph manipulations. We refer to [14] for a more comprehensive survey on using taxi GPS traces to analyze community dynamics where many studies use OD data derived from complete GPS traces. In this study, we align taxi pickup and drop-off locations to predefined zones in polygon datasets (e.g., Community District[14] in NYC) and use the polygon zones as nodes and the numbers of taxi trips as the weights of edges that connect the polygon zones. Clearly, different from road networks that planar graphs are formed purely based on geometry, non-planar OD graphs reflect the aggregated utilization of the underlying road networks.

To generate OD graphs and apply social network algorithms, it is necessary to utilize various spatial join techniques, such as point-in-polygon test based and point-to-nearest polygon based, to align OD locations with the underlying urban infrastructure, such as road segments and different types of zones. While large-scale spatial joins are computing intensive (as discussed before), generating OD graphs is an one-time cost and can be done offline, possibly by utilizing GPUs and/or computer clusters for accelerations. However, managing diverse OD graphs with complex node and edge structures and highly dynamic weights is technically non-trivial. As detailed in Section 3, we have chosen IBM SystemG as our graph database infrastructure to manage such OD graphs and to provide various analytical functionality through built-in and easy-to-use web-based APIs. Although we have chosen PageRank[15] as an example to demonstrate the feasibility from an integrated system development perspective in this study, we plan to incorporate more social network analysis functionality into the prototype platform by adopting and extending IBM SystemG modules. To the best of our knowledge, we are not aware of previous works on dynamically integrating Web-GIS and social network analysis functionality for visual analytics of OD data in a distributed (web) computing environment. Clearly such integration requires non-trivial coordination among Web-GIS frontend, spatial database backend and graph database backend. We discuss our designs and implementation details in the next Section.

3. System Architecture and Implementations

Our prototype platform currently consists of two backend servers and a frontend module to integrate services provided by the two backend servers as well as third party backends (Google Map services in particular). The high-level architecture is illustrated in Fig. 1 and the implementation details of the three components are provided in the subsections next. As the network/web communication and the GUI sub-modules are standard web and Web-GIS technologies, we will focus on the frontend geometry library sub-module that we have developed.

3.1 Geospatial backend

As illustrated in Fig. 1, the geospatial backend has a dual role: on-demand processing spatial (and spatiotemporal in the future) queries through client side visual exploration interfaces and offline aggregating OD records to generate dynamic graphs for online social network analysis and visualization. As the offline computation is not time critical and have been extensively studied in previous works, we next focus on on-demand query processing.

The most popular on-demand spatial query on OD data through a visual exploration interface in the current prototype platform is to retrieve the OD records (e.g., taxi trips in our NYC case study) whose origin or destination locations fall within an interactively drawn polygon, or the OD records that with the origin and the destination in a pair of interactively drawn polygons. Both require the point-in-polygon test based spatial operation which are well supported in traditional disk-resident Spatial Databases (e.g. [6]) and newly emerging hardware-accelerated prototype systems (e.g. [2]).

As discussed previously, our geospatial backend is designed to balance the tradeoffs between conventional database technologies, which are rich in functionality and easy to use but suffer from low performance, and hardware-accelerated geospatial query processing technologies, which are high-performance but are quite sophisticated and could be fragile due to lacking maturity. Different from traditional databases that typically follow row-based layouts for physical storage, we follow the columnar design and partition OD data based on both columns and rows for efficient data loading and subsequent in-memory processing (we refer to Section 3 of [2] for details). Given a limited number of interactively drawn polygons as user-defined ROIs, our geospatial backend scans all the OD locations and aggregates OD records that fall within the ROI polygons without relying on any prebuilt indices. A code segment using OpenMP[16] for straightforward parallelization is listed in Fig. 2. The approximately 20 lines of C code can be easily included in any C/C++ programs for simplicity and portability. Note that the 7 lines point-in-polygon test code is due to W. Randolph Franklin of RPI in 1990s[17]. We note that our GPU-based pint-in-polygon test module reported in [10] is based on the code as well.

Although we have not used any spatial index, we do use the MBR of a ROI query polygon represented by (*xmin*, *ymin*, *xmax*, *ymax*) to filter out location points that are outside of the MBR. Even for large ROIs where the MBR-based filtering is not effectively, scanning through all the yearly 170 million taxi pickup or drop off locations typically requires only a fraction of a second on an Intel dual quad-core machine running at 2.0 GHZ released in 2007. In contrast, a similar query against PostgreSQL/PostGIS database may take minutes. The query processing time using the simple technique could be even lower than network delays for web-based visual explorations. We found the performance satisfactory when the technique is applied to NYC taxi trip data at a yearly basis, although further refinement may be needed for larger-scale data. On the other hand, as the numbers of cores in multi-core machines are also increasing fast in the past few years [15], it is likely that the growth of CPU cores may match the growth of data volumes for the particular application that we are targeting at. This may further increase the applicability of the proposed simple technique.

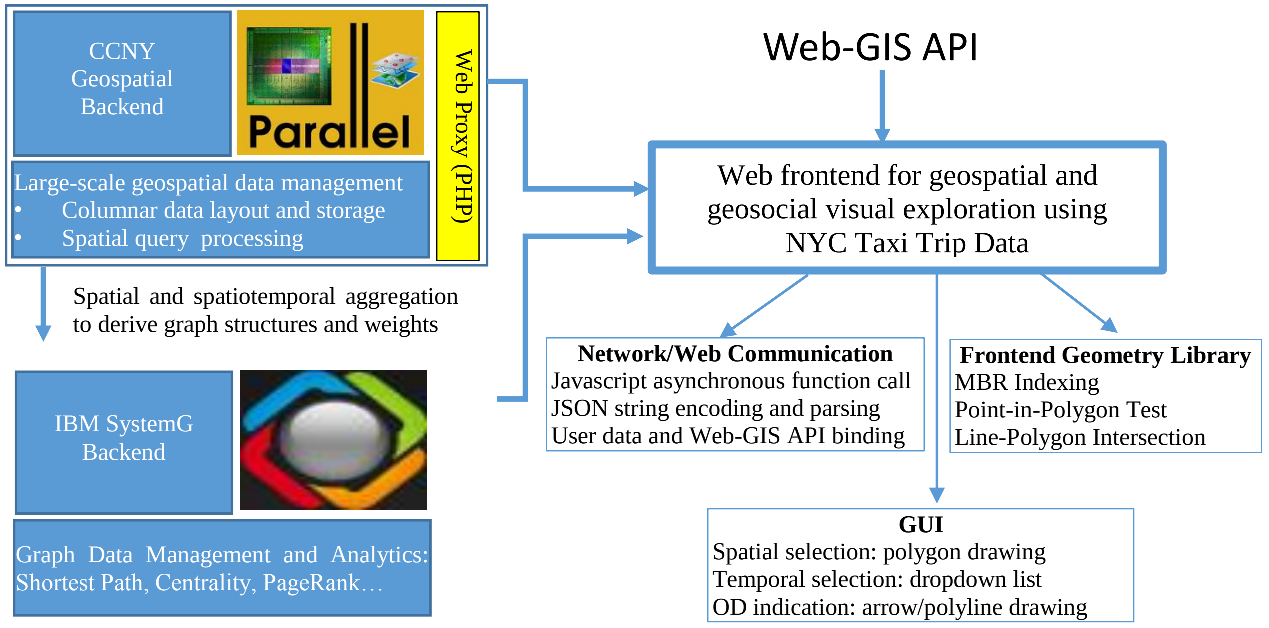

CCNY Geospatial Backend
Parallel
Web Proxy (PHP)
Large-scale geospatial data management
• Columnar data layout and storage
• Spatial query processing
Spatial and spatiotemporal aggregation to derive graph structures and weights
IBM SystemG Backend
Graph Data Management and Analytics: Shortest Path, Centrality, PageRank…
Web-GIS API
Web frontend for geospatial and geosocial visual exploration using NYC Taxi Trip Data
Network/Web Communication
Javascript asynchronous function call
JSON string encoding and parsing
User data and Web-GIS API binding
Frontend Geometry Library
MBR Indexing
Point-in-Polygon Test
Line-Polygon Intersection
GUI
Spatial selection: polygon drawing
Temporal selection: dropdown list
OD indication: arrow/polyline drawing

Fig. 1 Fig. 1 Prototype Platform System Architecture and Components

```
int pip_count(float vertices[][2], int num_vertices)
{
        …
        int count = 0;
        #pragma omp parallel for reduction(+:count)
          for (int i = 0; i < num_points; ++i) {
            double x = point_x[i];
            double y = point_y[i];
            if (x < xmin || x > xmax || y < ymin || y > ymax)
                  continue;
            bool in_polygon = false;
            for (int j = 0; j < num_vertices-1; ++j) {
              double x0 = vertices[j][0];
              double x1 = vertices[j+1][0];
              double y0 = vertices[j][1];
              double y1 = vertices[j+1][1];

              if ((((y0 <= y) && (y < y1)) ||
                  ((y1 <= y) && (y < y0))) &&
                (x < (x1 - x0) * (y - y0) / (y1 - y0) + x0))
                in_polygon = !in_polygon;
            }
            if (in_polygon) ++count;
          }
          return count;
}
```

Fig. 2 Code Segment of OpenMP based Parallel Processing of Interactive Spatial Query based on Point-in-Polygon Test

3.2 Graph Database for Social Network Analysis

SystemG is a graph database that is being actively developed at IBM Research, which explores efficient graph data organization for parallel computing architectures. SystemG is a whole spectrum solution for large scale graph processing, including graph storage, runtime, analytics and visualization [16]. In this study, we primarily use SystemG as a graph database backend to manage dynamical graphs and provide social network analysis functionality. As the possible combinations of spatial, temporal and thematic (e.g., zone datasets) selections from a visual exploration interface is considerably large, the backend needs to respond to such dynamic parameters during a visual exploration process, retrieve and transform the corresponding graphs, perform required graph analytics and send back the results to the client. As part of its analytics library, SysemG provides various social network analysis functionality, including shortest paths, betweeness centrality and PageRank. In this study, we will be using PageRank for demonstration purposes where graph weights are defined as the numbers of OD records between an OD pair, in a way similar to TrajGraph [13] where travel time between two OD zones is used as the weight of the corresponding OD graph. PageRank is useful in revealing the distributions of hot traffic hubs and the relative ranking of OD zones.

As an extension to SystemG's PageRank implementation, we consider not only graph structure (node degrees) but also edge weight, which is computed as the number of OD records (taxi trips) between an OD pair. The primary reason for the extension is that, for the NYC taxi trip dataset, the resulting OD graphs for pretty much all zoning systems are almost fully connected graphs. This is due to the close social-economic interactions among NYC zones. Without exploiting edge weight, classic PageRank algorithm designed for unweighted graphs will produce the same ranking scores for the zones which is not informative and is undesirable. Currently our prototype platform supports two zoning systems, i.e., Community District with 71 zones and Taxi Zone with 263 zones. The processing times of all the analytic modules that we have tested on the small graphs by SystemG is negligible. The high efficiency and high scalability of the graph databases backend allow much larger and much more complex graph analytics for better visual explorations and we leave it for our future work.

Another feature of SystemG we have exploited extensively is its built-in support for web-based applications. By starting the backend in a socket mode, all graph query and analytical results that are sent to terminals for debugging purposes in the interactive mode can be redirected to web clients. By setting the output format to JSON, the graph processing results can be easily consumed by web clients as Javascript objects and integrated with other web APIs, such as Google Map APIs for visualization.

3.3 Web Frontend

While many of the web frontend functionality utilizes standard techniques, such as Javascript asynchronous function call and JSON string encoding and parsing for distributed data communication and defining spatial parameters through polygon drawing (lower-right part of Fig. 1), in this subsection, we would like to highlight a few techniques that we consider unique to the prototype platform.

First, when querying OD pairs during an interactive visual exploration, after users draw both an origin polygon, a destination polygon and an arrow (all by Google Map APIs), we check the geometrical validity of the arrow at the web frontend by implementing the point-in-polygon test in Javascript. Only when both ends of an arrow fall within the two polygons, the web frontend considers it a valid OD pair query. Invalid OD polygon pairs will not be allowed to be sent to the geospatial backend to protect the backend from invalid queries and to lower its overhead.

Second, after a social network analysis by the graph database backend is completed and the results are geo-coded and visualized in the web frontend, we allow users to query graph weights of any OD pairs using a map interface. While the colored or patterned zones can show the distributions of the resulting ranking or centrality scores which can serve the purpose of "Overview" in visualization terminology [17], users may want to further look into the edge weights that are associated with certain nodes of the original graph to serve the purpose of providing "Detail", according to the well-known information seeking mantra – "Overview First, filter and zoom and details on demand" [17]. Although this can be easily implemented in a desktop computing environment, we found it non-trivial in a Web browser.

Our solution is to allow users draw an arrow on the map and determine the identifiers of the origin and the destination zones, again by applying the point-in-polygon test algorithm. Unfortunately, using Google Map APIs, while we are able to access the whole set of polygons of the base map that is being visualized, we are not able to get the active polygons in the current view and we have to perform the test on all polygons. While the number of polygons in a base map is typically small, the Javascript based geometry test is orders of magnitude slower than the backend side implementation and the performance is often unacceptable for interactive visual exploration. Again, we apply MBR based spatial filtering to limit the number of point-in-polygon test to solve the performance issue. We also observed that users are typically interested in nearby OD pairs which makes the query quite selective and the MBR based filtering highly effective in our experiments. An alternative might be to delegate the point-in-polygon test to server side. This would require the graph database backend to either communicate with the geospatial backend dynamically or implement the geometry operation inside the graph database backend. We are working on integrating the point-in-polygon test code originally developed for the geospatial backend into the graph database backend.

4. EXPERIMENTS AND DEMONSTRATIONS

Fig. 3 is a snapshot of a case study of querying an OD pair through the visual exploration interface. For the two interactively drawn polygons in the middle town region in NYC, it took the geospatial backend 136.29 milliseconds and 165.48 milliseconds to scan 173,179,763 pickup and drop-off locations in the two polygons and count numbers of locations that fall within the polygons. The performance is considered acceptable for interactive visual explorations. We are in the process of developing visual gadgets for temporal selection by learning from previous designs, such as TaxiVis [5] and OD-Wheel [9], and adapting for our web frontend.

Our second experiment is to demonstrate the utilization of the integrated Web-GIS and social network analysis for visual explorations of geosocial data. After users choose a certain zoning system (community district or taxi zone) and a weight metric (numbers of trips in hours 0-23 and their total) through dropdown lists, as shown in the top-left side of Fig. 4, the prototype communicates with the graph database backend and retrieves social network analysis results. While we currently uses colors to visualize PageRank results (red represents higher ranking and green for lower ranking), more sophisticated visualization techniques can be applied for better visualization. Form Fig. 4 it is clear that, taxi zones in the mid-town and downtown areas as well as the LaGuardia airport and the JFK airport regions (both are in Queens) are ranked much higher than those in Staten Island, which is expected and intuitive for non-expert users.

Assuming users are interested in retrieving the OD details between a taxi zone in the mid-town region and the LaGuardia airport region, as shown in the mid-left part of Fig. 4, they can draw an arrow to identify the original and the destination zones (polygons). If the original and the destination are validated, they will be highlighted (colored in yellow) and the detailed information will be requested form the graph database backend. Users can further switch to the interactive spatial query interface to draw polygons within or cross the boundaries of the predefined OD zones to obtain more specific OD information. While we are in the process to integrate the two web frontend interfaces, we hope the designs can serve as the starting point to better support seamless OD data explorations with both predefined zones and dynamically defined ROIs.

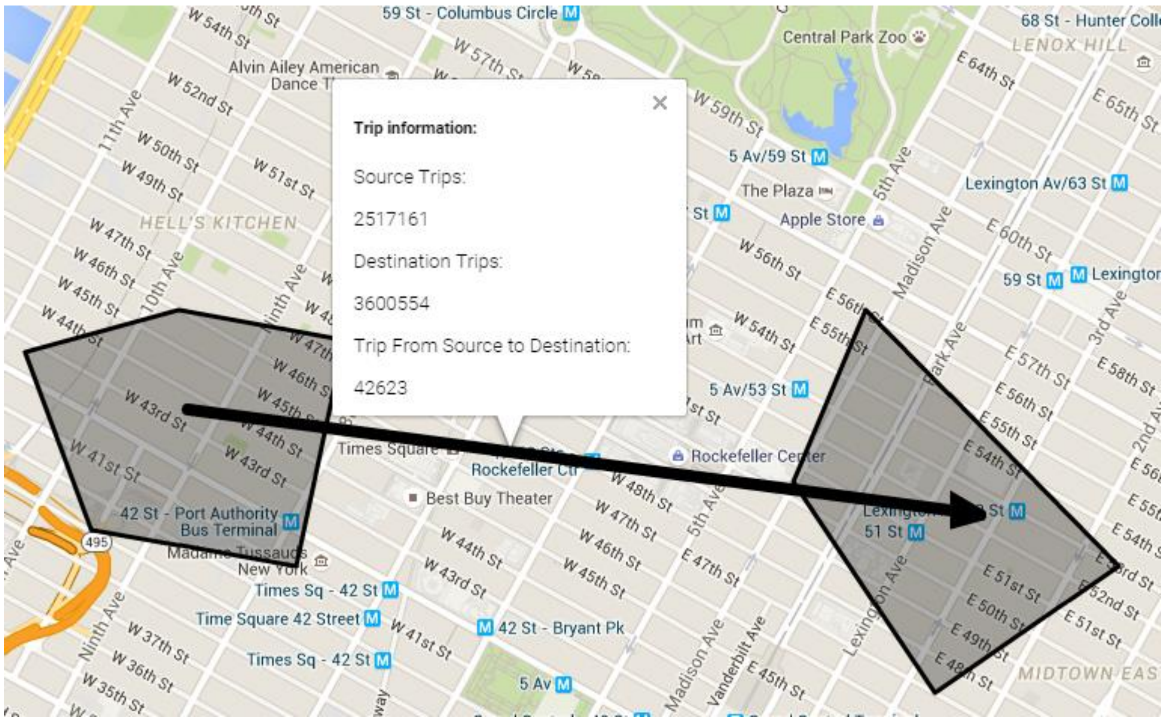
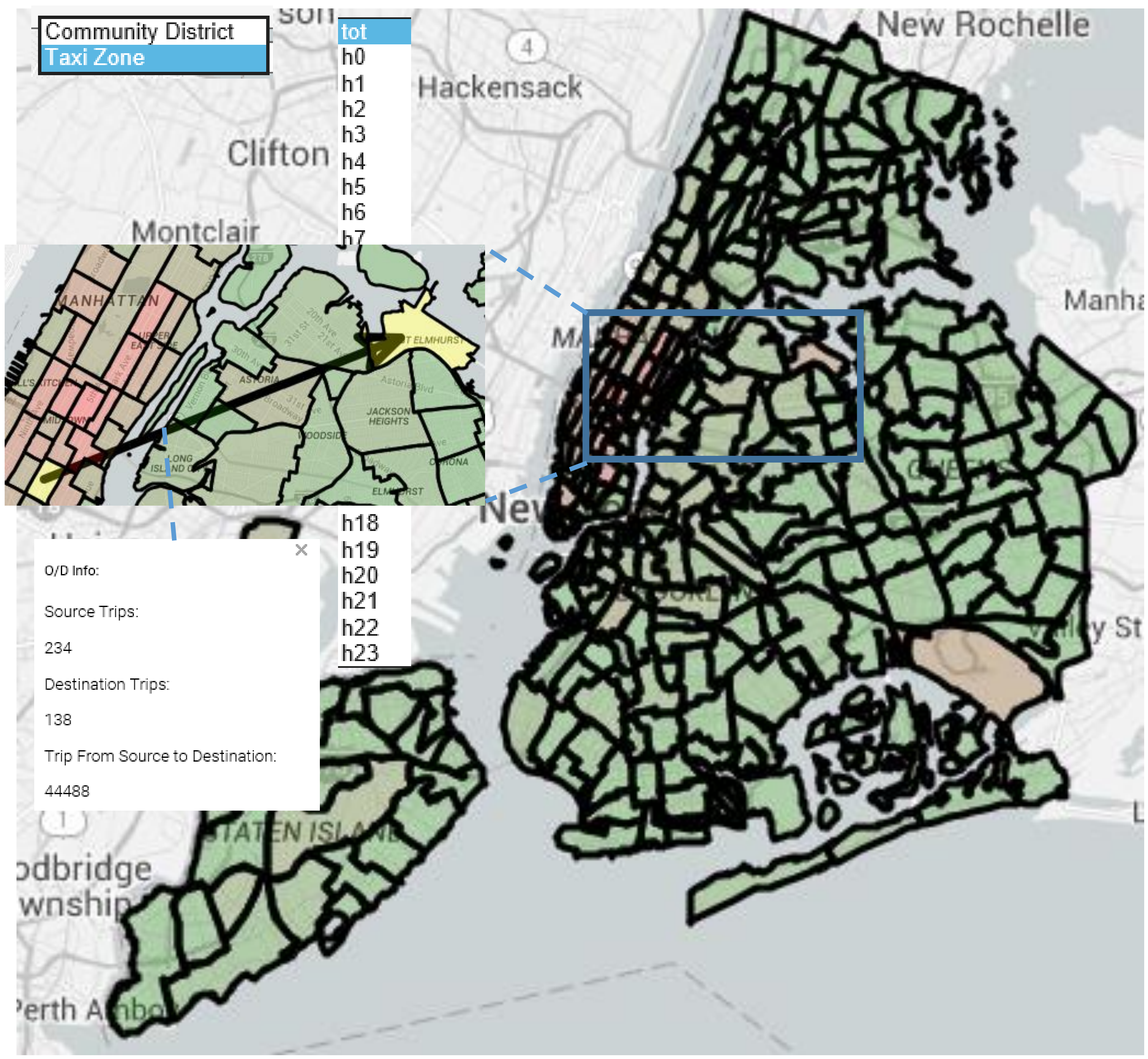

Trip information:
Source Trips:
2517161
Destination Trips:
3600554
Trip From Source to Destination:
42623

Fig. 3 Snapshot of an Interactive Spatial Query Processing Demonstration after Users Draw a Pair of OD Polygons



O/D Info:

Source Trips:

234

Destination Trips:

138

Trip From Source to Destination:

44488

Fig. 4 Visualizing and Exploring PageRank Results of Taxi Zones: Interfaces and Results

5. CONCLUSION AND FUTURE WORK

In this study, we report our work on developing a high-performance research platform to visually explore large-scale urban OD data in a web computing environment. Still under active development, the prototype platform integrates an in-memory parallel geospatial query processing backend and a graph database backend and provides several novel web frontend modules for both functionality and efficiency. Using the yearly 170+ million taxi trips in NYC, we have provided two experiments to demonstrate the utilization of an interactive query processing interface where users can define their OD ROIs interactively, and a geo-referenced social network analysis interface where graphs are dynamically defined, analytical results are visualized and details can be retrieved in an intuitive map-centric way.

The reported work is preliminary in nature and naturally leads to several future improvements. First of all, we plan to extend the geospatial backend to efficiently support more types of spatial queries, in addition to point-in-polygon test.

Second, as discussed inline, we plan to work with IBM SystemG development team to integrate spatial data processing functionality to support in-graph spatial queries. This may significantly reduce the coordination complexity among the web frontend and the two backend servers. Third, we plan to develop more intuitive visual gadgets for temporal selection in the web frontend, for both interactive spatial queries on raw OD data and social network inspired analysis on derived OD graphs.

Acknowledgement: This work is supported in part by NSF Grants IIS-1302423 and is part of a collaborative project between CCNY and IBM T. J. Watson Research Center under a Joint Study Agreement (JSA #W1463481).

6. REFERENCES

[1] D. B. Kirk and W.-m. W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, 2nd ed., Morgan Kaufmann, 2012.

[2] J. Zhang, S. You and L. Gruenwald, "Parallel Online Spatial and Temporal Aggregations on Multi-core CPUs and Many-Core GPUs," Information Systems, vol. 44, p. 134–154, 2014.

[3] J. Zhang, S. You and L. Gruenwald, "Large-Scale Spatial Data Processing on GPUs and GPU-Accelerated Clusters," ACM SIGSPATIAL Special, vol. 6, no. 3, pp. 27-34, 2014.

[4] E. H. Jacox and H. Samet, "Spatial Join Techniques," ACM Trans. Database Syst., vol. 32, no. 1, p. Article #7, 2007.

[5] N. Ferreira, J. Poco, H. T. Vo, J. Freire and C. T. Silva, "Visual Exploration of Big Spatio-Temporal Urban Data: A Study of New York City Taxi Trips," IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 12, pp. 2149-2158, 2013.

[6] R. Obe and L. Hsu, PostGIS in Action, Manning Publications, 2011.

[7] X. Jiang, C. Zheng, Y. Tian and R. Liang, "Large-scale taxi O/D visual analytics for understanding metropolitan human movement patterns," Journal of Visualization, vol. 18, no. 2, pp. 185-200, 2015.

[8] G. Andrienko, N. Andrienko, P. Bak, D. Keim and S. Wrobel, Visual Analytics of Movement, Springer, 2013.

[9] M. Lu, Z. Wang, J. Liang and X. Yuan, "OD-Wheel: Visual design to explore OD patterns of a central region," in 2015 IEEE Pacific Visualization Symposium (PacificVis), Hangzhou, China, 2015.

[10] J. Zhang and S. You, "Speeding up large-scale point-in-polygon test based spatial join on GPUs," in Proceedings of the ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data (BigSpatial'12), 23-32, 2012.

[11] J. Zhang, "Smarter Outlier Detection and Deeper Understanding of Large-scale Taxi Trip Records: A Case Study of NYC," in Proceedings of the ACM SIGKDD International Workshop on Urban Computing, Beijing, China, 2012.

[12] J. Zhang, "Efficient Frequent Sequence Mining on Taxi Trip Records Using Road Network Shortcuts," in Big Data Techniques and Technologies in Geoinformatics, CRC Press, 2014, p. 193–206.

[13] X. Huang, Y. Zhao, J. Yang, C. Zhang, C. Ma and X. Ye, "TrajGraph: A Graph-Based Visual Analytics Approach to Studying Urban," IEEE Transactions on Visualization and Computer Graphics, p. To Appear, 2015.

[14] P. S. Castro, D. Zhang, C. Chen, S. Li and G. Pan, "From taxi GPS traces to social and community dynamics: A survey," ACM Comput. Surv., vol. 46, no. 2, pp. 17:1--17:34, 2013.

[15] L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 5th edition, Morgan Kaufmann, 2011.

[16] Y. Xia, I. G. Tanase, L. Nai, W. Tan, Y. Liu, J. Crawford and C.-Y. Lin, "Graph analytics and storage," in IEEE BigData Conference, 2014.

[17] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," in Proceedings of the 1996 IEEE Symposium on Visual Languages, 1996.

---

[1] http://www.andresmh.com/nyctaxitrips/
[2] www.independent.co.uk/life-style/gadgets-and-tech/news/there-are-officially-more-mobile-devices-than-people-in-the-world-9780518.html
[3] https://developers.google.com/maps/
[4] https://github.com/ibmppl/ibmppl
[5] http://mapserver.org/
[6] http://geoserver.org/
[7] http://openlayers.org/
[8] http://www.esri.com/software/arcgis/arcgisserver
[9] http://www.opengeospatial.org/standards/gml
[10] https://developers.google.com/kml
[11] http://www.opengeospatial.org/
[12] http://www.opengeospatial.org/standards/wms
[13] http://www.opengeospatial.org/standards/wfs
[14] http://www.nyc.gov/html/dcp/html/neigh_info/nhmap.shtml
[15] https://en.wikipedia.org/wiki/PageRank
[16] http://openmp.org/wp/
[17] http://www.ecse.rpi.edu/~wrf/Research/Short_Notes/pnpoly.html