Real-Time Exploratory Visual Analysis Big Data

Abstract— Exploratory visual analysis is crucial to making sense of increasingly large data sets, sometimes with millions or more records. Designing an exploratory visual analysis system for real time interaction with big data is challenging. In this paper, we summarize the challenges of big data visual analysis systems from two aspects: 1) system architecture to support high performance and low latency for enabling real-time user interactions, and 2) interaction techniques to help analysts handle millions of items for dynamic querying. We propose our exploratory data analysis system: AVIST – a GPU based animated visualization toolkit targeting large time-series, multi-dimensional data sets. Our first contribution is a GPU based in-situ visualization architecture for data processing running simultaneously with visual rendering, which maximizes GPU parallel computing capability and reduces I/O costs for data transformation. To further improve the performance, we propose a data dependency graph design for charactering the multi-stage of data transformation. We employ user interaction techniques to trigger the dependency graph. Those interactions such as animation, associated data filters, brushing and linking can help analysts quickly get insights from data. At last, we demonstrate that our system can easily identify abnormal behaviors and infer hypothesis in two case studies.

Index Terms—Big data, interactive data exploration and discovery

1 Introduction

Visual analysis (VA) is "the science of analytical reasoning facilitated by interactive visual interfaces" [30] and visual analysis systems are keys to supporting analysts for sense-making, reasoning and decision making. Previous visual analysis systems focus more on novel visualization and interaction techniques. As more and more data are generated at high volume and with increasing complexity, the solutions for scaling VA systems to big data require further investigation [34]. While the definition of "big data" is debatable, in this paper we follow an existing definition from previous research and use one million or more data items as a threshold [22].

Interactively visual exploration is important for insight discovery of big data. When faced with big data, analysts may not lost in information see. Exploratory analysis is a way for "unknown-unknown" knowledge discovery: exploring data, formulating possible hypotheses, and exploring data again for verification or new hypotheses generation. In this paper, we focus on designing a VA system for exploratory analysis of big data. Specifically, we target the high volume of time-series and multi-dimensional data. To handle such data, we present AVIST: a GPU based animation visualization toolkit. We hope that AVIST can assist cyber security analysts find potential threats from big router logs and help financial analysts to analyze custom transactions to provide better services.

Research about big data exploratory visual analysis systems should answer two key questions: 1) system design for handling big data to support real time interaction, and 2) interaction and visualization techniques to facilitate big data interpretation. The former question is related with the data processing techniques, while the latter concerns the scalability of visualizations and interactions. In this paper, we carefully consider these two questions and present AVIST for real time visual exploration of big data.

GPU based In-Situ Visualization Architecture. Previous researches tackle the big data problem from the data reduction and compression aspects, spent tons of time in data preprocessing to generate related statistics information for users' exploration. While we admit their contributions for data sampling, aggregation, and modeling, we argue that the raw data is more important and attempt to prevent losing information during the data processing stage, which is the core part for exploratory data analysis. Our paper aims at helping analysts to find a needle in haystack, rather than telling them the shape of the haystack or the distribution of the needles. In this paper, we provide a GPU based in-situ visualization architecture, which stores all raw data in GPU memory and features the data processing and visual rendering simultaneous based on the powerful GPU parallel computing performance.

Big data visualization and interactions. Visualizing millions of items in a given resolution of display (1∼3 million pixels) may have perceptual and interactive problems [22]. We agree that visualizing all data points may lead to over-plotting and may overwhelm users' perceptual and cognitive capacities. However, most visual clutter problem happens due to lacking of efficient interaction techniques. We argue that visually presenting all data items and providing powerful user interactions for filtering large data can help users to perceive more. For exploratory analysis, over-plotting may indicate some kind of information that users need to perceive and may direct users' attention to next interaction and hypothesis . In AVIST system, we incorporate animation, combined data filters and brushing and linking interaction techniques to facilitate users to explore large data and enable them to find a needle in haystack. Besides, we carefully design the data flow in AVIST and present a data dependency graph to incorporate those interactions for data transformation.

In all, we present the AVIST system for real-time exploratory visual analysis of big time-series and multidimensional datasets. The AVIST system features a GPU based in-situ visualization architecture for quickly mining big data based on user interactions, which enables users to quickly identify a needle in a haystack. We use two cases to demonstrate the features of our system.

2 Related Work

Interactive big data visualization and analysis systems are related to lots of research areas. In this section, we present our survey from three aspects: 1) system design and data processing for handling big data, 2) visualization and interaction techniques that can scale to big data, and 3) parallel computing for big data visual analysis.

2.1 VA Systems and Data Processing

Companies have already released their VA systems for the growing market. *Tableau* [4] is the pioneer and it can handle big data based on its Tableau Data Engine(TDE) [32], which is a specialized column-oriented store modeled after MonetDB [7]. *QlikView* [2] is also a major player regarding in-memory architecture to support interactive drill-down capabilities. *Spotfire* [3] is awarded for its automatic analytics, interactive visualization, system architecture and data management. Zhang et al. [34] survey the state-of-art commercial VA frameworks(e.g. *Tableau* [4], *QlikView* [2], *Spotfire* [3], *JMP* [1]), and point

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

that there is still room for improvement and a number of challenges for future directions such as real time analysis, predictive analysis and so on.

Actually, lots of commercial products have roots in academic research such as *Tableau* [4] from Stanford University and *Spotfire* [3] from University of Maryland. Academic researchers are also active in big data visual analysis system. In the database community, researches about in-memory databases and data cubes are becoming the foundations for big data systems. Recent big data visual analytics systems such as Nanocubes [21] and imMens [22] are based on data cubes. SAP designs Hana-DB [12], which is an in-memory database system for business analytical applications. Besides, data aggregation methods are also investigated. imMens [22] emphasizes binned aggregation. Jugel et al. [19] propose a new visualization-oriented data aggregation technique for visualizing big time-series data. Moreover, Query performance is an important issue for VA systems. Agarwal et al. [5] propose BlinkDB, which allows users to trade-off query accuracy for response time. Fisher et al. [15] also propose a similar idea about approximate database queries for exploratory data analysis. VisReduce [18] is an incremental computing visualization system that is based on a modified Map-Reduce-style algorithm.

In the visual analytics field, researchers published the methodology of the visual analysis system. Fekete et al. [13] emphasize VA systems on three important layers: visualization, analytics and data management, and it analyzes the main issues of each layer to support interactive exploration of big data. Morton et al. [25] present a vision of next-generation visual analytics services covered three related capabilities inspired by the sense-making model. Stolper et al. [28] give a progressive visual analytics model which enables an analyst to inspect partial results without sacrificing computing speed. Battle et al. [6] present a three-tiered visualization system for automatic reduction of the query results.

2.2 Visualization and Interaction

Big data visualization often results in occlusion and cluttering problems. Researchers in information visualization area have proposed many clutter reduction techniques. Ellis et al. [10] give a taxonomy of clutter reduction techniques which can be divided into three categories: appearance, spatial distortion and animation. Besides, Fekete et al. [14] propose several non-standard visual attributes such as stereovision or synthetic overlap count to enhance visualization.

Another challenge for big data visualization is the human interaction techniques to facilitate users quickly identifying patterns and outliers. Fekete et al. [14] summarize the interactive techniques from four aspects: space multiplexing, time multiplexing, overlapping and space deformation. Space multiplexing is displaying two or more visualization configurations on the same screen while the time multiplexing technique shows each configuration successively at regular pace (animation) or by iterative methods such as dynamic queries. Overplotting shows transient information over the visualization and space-deformation only shows important features based on sampling and aggregation. Cross filtering [31] is also an important interaction technique for exploring big data with high dimensional attributes which supports expression of complex multidimensional queries using simple interactions.

2.3 Parallel Computing

Parallel computing and distributed systems are becoming significant parts of big data visual analysis systems. Google has designed Dremel [24], a distributed system for interactive analysis of large scale nested data. Based on Hadoop, Starfish [17] is a self-tuning system for big data analytics. It adapts to user needs and workloads to provide better performance. Chen et al. [8] survey Map-Reduce based systems for interactive query processing of big data and point to new design directions for shifting from long-running batch jobs to interactive analysis.

GPUs are designed for videogame and graphics but their remarkable development in performance and programmability makes them popular for general purpose computing. And a number of researchers have proposed GPU based solutions for big data analysis. Fekete et al. [14] emphasize using accelerated graphics hardware to provide high-density interactive visualization. imMens [22] supports visual query of big data based on parallel processing. Zinsmaier et al. [36] utilize GPU computing for real time interaction with million node graphs.

3 Design Analysis

We emphasize real time visual exploration of big data. We argue that such VA systems need to support users **finding a needle in haystack**, which means that VA systems should provide **real time ad hoc query and drill down filtering** of big data to help user find outliers and identify pattens. Based on these, we summarize and analyze previous techniques for designing VA systems.

3.1 Data Processing

First, we focus on data processing. We want to achieve both high performance and support for ad hoc querying in our VA system.

In-memory database is a popular technique in commercial big data VA products. Compared with traditional on-disk databases, in-memory databases are faster at accessing data by eliminating disk seek time, which guarantees faster and more predictable performance. While the memory capability constrains data size, solutions such as hybrid database systems or distributed in-memory architectures [20] are directions for future research.

Data cube is an aggregate operator for On Line Analytical Processing (OLAP) in the context of data warehousing. However, cube size explodes as data dimensionality increases, which has a significant storage requirement even than original datasets. Also, data cube needs heavily pre-computing for aggregation, and do not easily support to querying down to individual record.

Approximated and incremental queries are promising techniques for big data analysis. Fisher et al. [16] show the utility of the approximated queries by interviewing three teams of analysts, and conclude that incremental query interactions for data analysis is tractable and desirable. Analysts can obtain immediate feedback by incremental queries to refine their results, even further to explore new avenues.

Data reduction refers to general methods for dealing with big data, including filtering, sampling, aggregation and modeling. Liu et al. [22] give a survey about data reduction techniques, and they argue that sampling strategies and model based abstractions require prior knowledge and preprocessing of big data, while filtering and aggregation techniques are more effective in some cases. They adopt data aggregation technique to generate data cubes for real time visual query. However, this paper cannot afford drill-down filtering.

In all, we summarize major techniques for database design and query processing in table 1, which are the guides for designing VA systems. Our paper aims to support ad hoc query and drill down interactions, data cube technique is not our choice. Data sampling and modeling need prior knowledge of big data, which are also out of our options. So we adopt the in-memory database, incremental queries, filtering and aggregation methods for designing VA system.

Table 1. Database Design Analysis

Technique		Pros	Cons
In-memory databse		Fast performance	Memory size limitation
Data cube		Fast performance Aggregation query	Dimension scalability Lack of drill down query pre-computing
Incremental Query		Fast performance	Approximated query
Data Reduction	Sampling Modeling	Fast performance reduction	Prior-knowledge pre-computing
	Filtering	ad hoc query	in-place computing
	Aggregation	Fast performance reduction	Lack of drill down query pre-computing

3.2 Visualization and Interaction

Second, we explore existing big data visualization and interaction techniques. Especially, we focus on visualization and interaction of large high dimensional datasets.

3.2.1 Visualization

Coordinated and multiple views (CMV) [26] are widely used for exploratory analysis. They allow users to see the data in various forms, to manipulate the visual presentation in different ways and to interact and coordinate the interactions between different views. High-dimensional datasets include multiple data attributes, which may need various visual presentations, and CMV are appropriate design for visualizing high dimensional datasets.

Scatterplot matrix (SPLOM) [11] is a well-established technique to visually explore high-dimensional data sets. It arranges the scatter plots of all possible combinations of dimensions to visual the data. So the number of scatter plots quadratically grows with the data dimensions and the drawing area becomes narrower. Over-plotting happens by increasing the number of points, while data analysis becomes much more difficult.

Parallel coordinate plots (PCPs) are a widely used technique for visual analysis high dimensional data sets which maps each point as a polyline among the parallel axes. It scales very well with the number of data dimensions. However, it may suffer the visual clutter with the rapid growth data size. Another consideration is the ordering of axes, which may heavily affect the visualizing results.

While we admit that we may not list all of the visualization techniques for big high dimensional data sets, these three are the most important methods. In our VA system design, we favor coordinated and multiple views (**CMV**), and parallel coordinate plots (**PCPs**) instead of scatterplot matrix.

3.2.2 Animation and Interaction

User can control the visual items by animation and interaction techniques. We list three common animation and interaction techniques for exploratory data analysis.

Brushing and Linking is used to explore tightly coupled data relationships by highlighting items in multiple views. imMens [22] is an example to highlight brushing and linking technique for visual query big data. Cross-filtering [31] generalizes the brushing and linking technique into a design pattern for fast and flexible visual drill-down into fine grained relationships in multi-dimensional data sets.

Animation is viewed as special filter based on time, and it automatically updates data items between frames. During the frame transitions, temporal patterns may be revealed. Compared with brushing and linking, which is a space multiplexing technique, animation is a time multiplexing technique for visualizing data items at a regular pace.

Dynamic Query means interactively filtering and redisplaying of the data items through successive interactions. Brushing and linking, animation and widgets as "range-silder" are the general ways for dynamic queries. To explore the high dimensional datasets, more complex filters and queries need be investigated. Zhao et al. [35] present a visual query language in PivotSlice to identify implicit and explicit relationship. Polaris [29] allows users to drag and drop visualizations based on its table algebra.

Other interaction techniques may also be effective for big data visualization. We believe that the listed three are crucial for exploring big data. We want to incorporate these techniques in our VA system to achieve dynamic querying to help users drilling down each record.

3.3 Parallel Computing

This section gives a brief discussion of the distributed systems and parallel computing techniques for big data visual analysis.

Distributed systems are becoming a hot topic for big data visual analysis, and many distributed VA systems have already deployed, such as Google's Dremel [24]. However most of them are based on Map-Reduce scheme [9], that they are designed for large, long-running bath jobs and are incapable of many small, short and increasing interactive queries [8]. To make best use of distributed systems, Dremel [24] contributes a novel data model, which is a columnar representation of nested data for dynamic query. Moreover, Starfish [17] characterizes the work load of distributed systems and applies optimization for scheduling jobs to achieve better performance. However, such kinds of improvements focus on shipping the computation to a cluster of processing nodes, while shipping large data in the cluster increases I/O costs, which may not achieve real time performance for interaction with big data.

GPUs are another potential solution for analyzing big data. Besides their powerful rendering capability, GPUs feature the general purpose computing. For example, Fekete et al. [14] suggested utilizing GPU rendering power to achieve the redisplay speed required for smooth interaction of big data. When user triggers a series of queries, all data items are send to the GPU for fast rendering. imMens [22] makes a further step for visually querying big data based on GPU parallel computing. However, GPU based solutions suffers limitations, such as limited GPU memory size, copying data back and forth between CPU and GPU, which increases I/O costs.

Generally, parallel computing is promising for big data VA system and both distributed systems and GPUs are the feasible solutions. In our system design, we leverage the power of GPUs for fast rendering and parallel computing. We acknowledge the shortcomings of GPUs, and try to overcome them to achieve good performance. Table 2 shows the summary of all mentioned techniques we try to incorporate into our VA system.

Table 2. Summary of VA system techniques

Database	Visualization	Interaction	GPU
In-memory Incremental Query Filtering Aggregation	CMV PCPs	Brushing & Linking Animation Dynamic Query	Fast Rendering GPU Computing

4 GPU BASED IN-SITU VISUALIZATION ARCHITECTURE

Wong et al. [33] list top 10 challenges in extreme-scale visual analysis and *in situ* analysis ranks the top one. Compared with traditional approaches, in situ VA tries to perform as much analysis as possible while the data are in memory to reduce I/O cost. In scientific visualization, in-situ processing and visualization are the general methods for handling big data [23], especially when I/O becomes the performance bottleneck. The idea is that simulation and visualization code run simultaneously which can reduce the data transfer and storage costs. In this paper, we adopt *in-situ* concept, and propose that **data processing and visualization code can run together to reduce the number of visual primitives transferring between GPU memory and main memory** and that all code can run on the GPU to fully utilize GPU parallel resources.

Figure 1 shows our GPU based in-situ visualization architecture. Our architecture also incorporates previous mentioned visualization and interaction techniques. In the CPU sides, we handle only user interaction techniques which trigger GPU parallel processing and rendering. The GPU controls all data transformation from raw data into the visualizations. It can process related queries and generate visual primitives simultaneous. All raw data is located in GPU memory to support users' drill-down query of each data record. All generated visual primitives are stored in GPU vertex buffer objects (VBO) to obtain substantial performance gains and avoid data transformation with main memory.

Our in-situ visualization architecture makes the best use of GPUs powerful computing and rendering performance. It supports filtering and visualizing items in parallel. More importantly this kind of design avoids data transformation between main memory and GPU to reduce IO cost, which is potential bottleneck for big data VA systems.

However, the major consideration of the in-situ visualization architecture is data scalability problem which is limited by GPU memory capability. To feed more data into the GPU memory, we propose to preprocess datasets into compressed binary format.

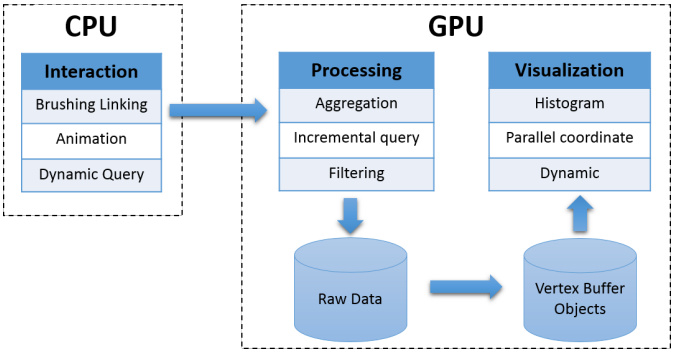

Fig. 1. The *In-Situ* visualization architecture of the *AVIST*.

Considering general multi-dimensional datasets, we first identify each data type in datasets. For categorical and ordinal data, we count all possible values and map each value into a unique ID. All values and their corresponding IDs are stored in main memory as meta-data. Then we store datasets in GPU with row-oriented format, replace their original value to its ID in binary format of one byte, two bytes or int (e.g. if one dimension data only has less than 256 possible values, we can use one byte for representation.). For quantitative data, we simply store one *Int* or *Float* for representing each data value. We store time data in *time_t* format, which occupies 8 bytes. Table 3 summarizes our data preprocessing strategy. The row-oriented design targets temporal and spatial locality based analytical tasks for drilling-down query, supports analysts to find a "needle in a haystack". The preprocessing targets analyzing each column for compression and making sense of the whole data.

Table 3. Data preprocessing strategy

DataType	GPU memory binary format	Main memory meta-data
Time	time_t 8 bytes	minimum maximum
Quantitative	Int or Float 4 bytes	minimum maximum
Categorial Ordinal	$1 \sim 4$ bytes	Dictionary (id and data value)

A standard commercial graphics card has its memory from 4GB to 8GB. Considering a dataset with most of its data attributes occupy 4 bytes, then the total data items we can handle is about 10 millions. So our VA system can handle million records based on one GPU.

5 Visualization and Interaction in AVIST

In AVIST, coordinated multiple views are provided for exploratory multi-dimensional data analysis. Brushing and linking, animation and dynamic queries are deployed in data views. To further improve performance and incorporate with incremental querying and animation, AVIST features a data dependency graph for charactering data transformations, which is also an instantiation of cross filtering design pattern [31].

5.1 Visualizations and Interactions

Three data views are provided: histogram view, parallel coordinated view and dynamic view. Figure 2 shows these three data views with a control panel. Other data views can also be flexible added in our architecture. For example, we add a virtual globe view for spatial data visualization in our case studies.

Histogram view shows the data distribution of current time window. Users can select different dimensions from a listbox to explore different aggregation information.

Parallel coordinated views show the detail of each data record. Users can select multiple data dimensions for generating their custom parallel coordinate plots. The axes can be re-ordered based on users' selected order in the listbox.

Dynamic view is a time series chart, which shows the data aggregation of certain filtered events over a period of time. When a user change data filters, the dynamic view clears previous visual primitives and re-draw everything.

Control panel provides the user interactions and animation. By playing the animation of the data at certain pace, hidden temporal patterns can be revealed. The animation control supports automatic forward playback, interactively dragging of the time window bar, and interactive change of animation speed. With changes of the current time and time range, users can define a time window in which the information will be analyzed and visualized at the correlated views. By combining automated animation with these correlated views, AVIST provides temporal changes in the datasets, which affords discovery interesting temporal patterns for further analysis.

Besides the time window for data slicing, AVIST features the combined data filtering. Three different filters are implemented: 1) highlight filters which make users' selected values stand out of the rest data with different colors, 2) exclusive filters to remove non-interesting data from data views and 3) negative exclusive filters, the exact opposite of exclusive filters, which remove all data except items marked interesting by a user. Each kind of data filters have three modes: 1) *Solo mode* , which allows only one filter value in current filter set, 2) *and mode* to combine several filters together as a complex filter emphasizing that data records need satisfy all requirements for visualization and 3) *or mode*, which means that data records just need to meet only one of the filter requirements.

Based on these three basic filters with their three modes, users can nest them to generate complex data filters which help to drill down each piece of data record. These filters with the brushing and linking interaction technique can quickly help users to explore datasets. In Figure 2, users select highlight filters with *or mode*, and then brush their selected colors in *firstSeenDespIp* axis of parallel coordinated view to identify three suspicious IP addresses. The histogram view shows the distributions of *ipLayerProtocol*, which shows the highlighted activities happen in TCP flows. The dynamic view shows the highlighted transactions are a periodically behavior in the network.

5.2 Data Dependency Graph

We employ the cross filtering design pattern [31] in our GPU based in-situ visualization architecture, which contributes a data dependency graph for organizing the data flow on the GPU.

Figure 3 shows our data dependency graph, which is separated into three parts. Firstly, the top level graph shows the user interactions of CPU side. Users can manipulate four filters based on their interactions: *time windows*, *exclusive filter*, *negative exclusive filter* and *highlight filter*. All filters are passed into GPU for data processing. The middle part of the graph is the GPU data parallel processing. The raw data is filtered by *time windows* first, then both *exclusive filter* and *negative exclusive filter* are applied to remove filtered data items. At last, *highlight filter* is deployed to generate highlight datasets. Next, there are two steps that they transform original data items into visual primitives in each data view. The first one generates geometry data which are values of visual primitives. The second step is about rendering data, and it scales visual primitives into 2D screen considering data ranges. At last, all rendering data are passed into GPU VBOs. This kind of design makes that all data views share the raw data, filtered data and highlight filtered data, while each data view has its own geometry and rendering data. The last part in this graph is GPU rendering, which transforms all visual primitives into pixel colors.

Our data dependency graph has several advantages. First, it follows the cross filtering design pattern [31] and three filters with three modes supports the drill down query of multi-dimensional datasets. Users interactions trigger the data flows to update the visualizations. Second, all data processing and visual rendering can fully utilize GPU resources to achieve best performance. Third, incremental computing

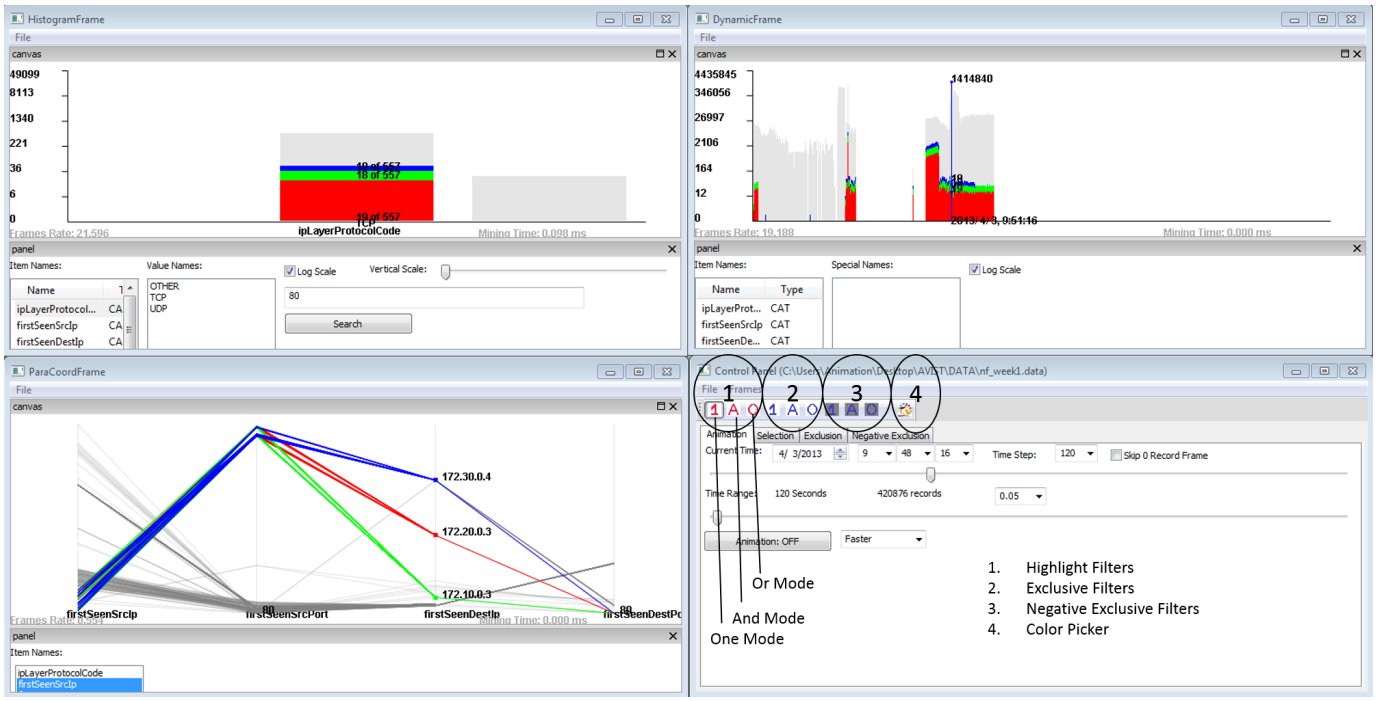

Fig. 2. Three data views and control panel in AVIST system. The top left is histogram view, top right is dynamic view, bottom left is parallel coordinated view and bottom right is control panel. Each data view is separated into two parts: the canvas for rendering visual primitives and the panel for exploring different data dimensions.

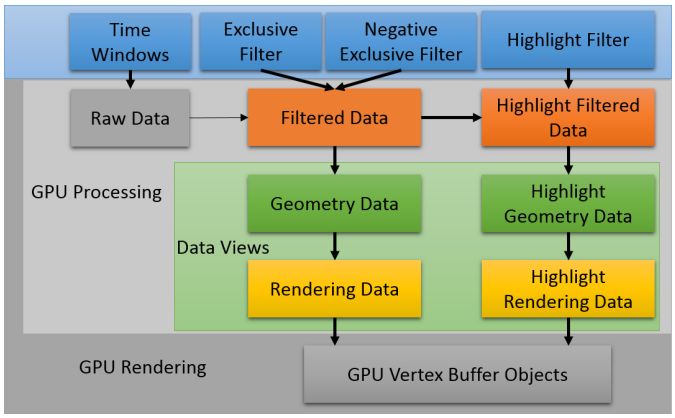

Fig. 3. The data dependency graph of AVIST system.

can be applied. When users play animation, frame-to-frame coherence can be exploited. In such cases, only incremental data need to be considered. Besides, user interactions also follow incremental format for dill-down queries. If a user perform *or* operator of highlight filters, previous highlighting visual primitives do not need to be recomputed. When performing *and* operator, incremental computing is only applied in previous highlight data which trigger the right column of data graph, while all other data are still the same. In all, we formalize the computation flow as a direct acyclic graph (DAG) of tasks, fully incorporated user interactions and incremental computing.

6 Implementation Performance

6.1 Implementation

The AVIST system is written in C++, its computation codes are based on CUDA 5.5 and visualization codes are based on OpenGL and GLSL. The interface are coded by wxWidgets. We use the Thrust library[1] for accelerating sort, scan, transform and reduction operations.

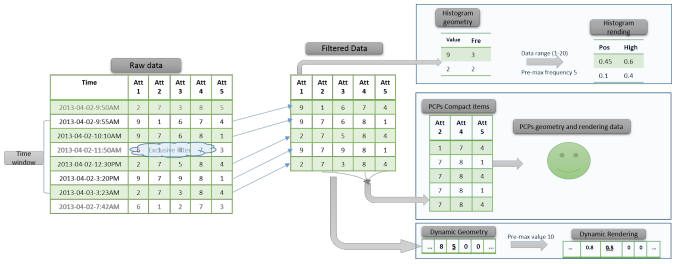

Fig. 4. This figure shows data transform for generating geometry and rendering data for histogram and dynamic view.

Figure 4 demonstrate the transform from raw data into histogram and dynamic views. Firstly, time window is applied to raw data by searching the start and end time point to slice current chuck of data, then each data record is validated by filters in parallel to compact the filtered dataset. Based on user's chosen data dimension in histogram view, GPUs group the corresponding column data to generate geometry data and transform it to rendering data based on the meta-data (data range). The data flow in dynamic view is straightforward. GPUs summarize the filtered data items for geometry data, then transform it to height value based on previous maximum value.

Data transformation in parallel coordinate views is more complex than other views. Firstly, GPUs generate PCPs compact items based on user chosen axes. Then the data flow is separated into two branches for parsing vertex and edge data. To generate vertexes information, an unique operator is applied to each column and then all unique values are organized into *vertex* array with *offset* array. Then the data ranges are applied to geometry data to obtain each node's position. For edges,

[1] https://developer.nvidia.com/Thrust

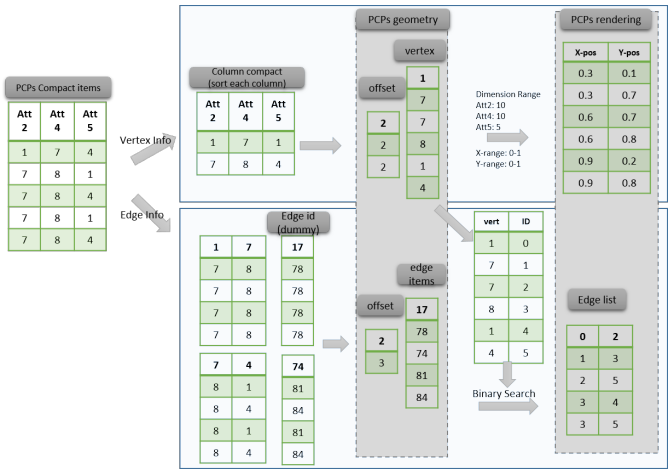

Fig. 5. This figure shows data transform for generating parallel coordinated views.

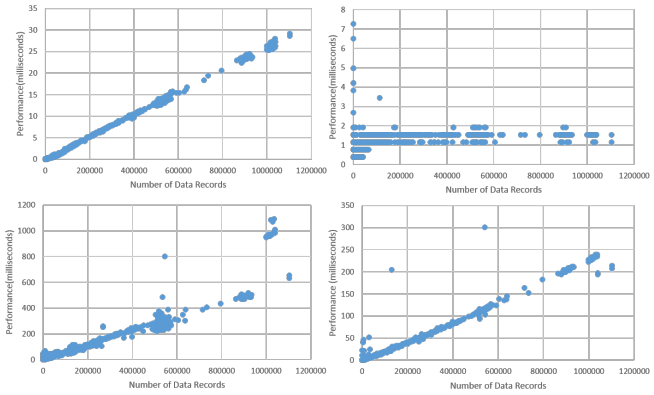

Fig. 6. The performance of AVIST system. These scatter plots show the relationship between number of queried data records and the performance (milliseconds). From the top left to the bottom right are the performances of histogram view, dynamic view, parallel coordinated view and the filtered data.

every two neighboring column are combined into an array by their dummy values, then they are compacted and organized into *edge* array and *offset* array. After this, the dummy values are unpacked, which are separated into two columns for representing edge list. The edge list replaces each vertex ID with its order ID for generating rendering data.

6.2 Performance Analysis

We test the performance of AVIST system on a desktop computer, running Windows 7 Enterprise with Service Pack 1, which is equipped with an Intel i7 processor and an NVIDIA GeForce GTX 680 graphics card with 4GB memory. And we use the network traffic dataset from case study one as the benchmark to profile AVIST system.

Based on our data dependency graph, we character the AVIST system in two stages. The first stage is about filtered data generation. This stage is shared by all data views. The second stage is about each data view visual primitives generation, and their performance are independent with each other. Figure 6 shows the performance in scatter plots. We see that the performance has a linear relationship with the queried data records except the dynamic view. Actually, the aggregated information in dynamic view can be easily derived by filtered dataset. The performance of parallel coordinated view is the bottleneck in AVIST system, because it generates a number of visual primitives to match each data record. If the number of queried records exceeds 600,000, AVIST can not afford real time animation and interaction due to the hardware computation limitation. This also suggests that user should filter more data in current time window or shrink his time window size to drill-down query the data details.

7 User Cases

Below we describe two examples AVIST explorations of large volume of time series and multidimensional datasets. We show how data patterns are revealed by animation and drilling down queries and show how AVIST system is applied for large spatial-temporal data analysis.

7.1 Network Flow Analysis

Exploratory visual analysis network logs is critical for detecting potential cyber threats and network intrusions. VAST 2013 Mini Challenge 3, which targets the analysis of Big Marketing company's network, provides big network traffic and a variety of logs[2]. Among all the datasets, the biggest one is about week one network flow, which includes 46,138,310 records with 16 dimensions, so as the total data items is about 738,212,960.

[2]http://vacommunity.org/VAST+Challenge+2013

Analysts first preprocess original dataset, and obtain the compressed binary data with 1.8GB. Then analysts load the data into AVIST system to explore the network traffic and to identify the potential threats. Figure 8 shows one usage scenario to demonstrate how analysts find potential network security threats and detect temporal patterns based on AVIST system.

First of all, analysts specify animation speed and time window size (120 seconds in this example) to play animation. The overview of network traffic is visualized in dynamic view, shown in subgraph 1. Analysts then identify an unusual behavior that there is no any network flow from 4-2-2013 9:40 am to 4-3-2013 3:26 am. To investigate the details, analysts generate the parallel coordinate view in subgraph 2 by choosing data dimensions in order of *firstSeenSrcIp*, *firstSeenSrcPort*, *firstSeenDestIp* and *firstSeenDestPort*. Analysts then play animation again and discover an interesting visual pattern. Analysts character this pattern by highlight related data records. In subgraph 3, they learn that the destination ip 172.30.0.4 was accessed by four source IPs 10.10.6.2, 10.6.6.6, 10.7.6.5, 10.7.7.10 from their all available ports during 4-2-2013 5:20 am. However, analysts find that the parallel coordinate view are always over-cluttered. Subgraph 4 shows the network traffic distribution based on *firstSeenDestPort*, and analysts realize that most of the network records are related with port 80. They first assume that these records may be normal and they want to remove these data records to see buried patterns, so they apply exclusive filters to remove these network logs visualization in subgraph 5. Then they discover a port scan behavior from parallel coordinate view in subgraph 7, and they character this pattern by applying different colors in each highlight filter, shown in subgraph 6. The whole picture of port scan behavior is shown in Figure 2. Analysts play and play back animation, and identify that the port scan only happens in TCP flow and it may be a periodical behavior. All of those information can guide analysts next visual query interaction to infer more.

This scenario shows that AVIST system can facilitate analysts incremental and dirll-down query for exploratory analysis the big network traffic. The combined data filters can help analysts easily to remove visual clutter and to character abnormal patterns. By playing animation, analysts can identify temporal behaviors to infer more.

7.2 International Trade Analysis

Making sense of international trade is important for government and police makers. With the world economic growth, world trade transactions become large and complex. In this case study, we obtain big international trade transactions from PIERS Global Intelligence So-

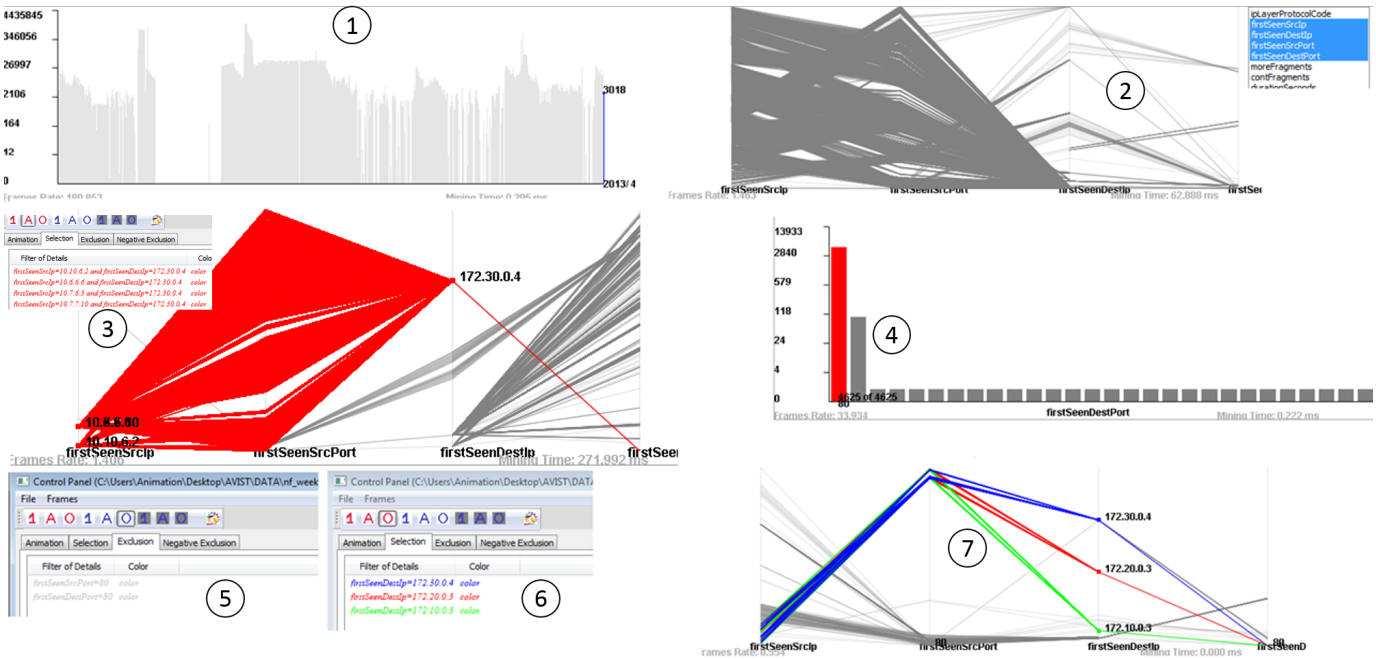

Fig. 7. This figure shows a usage scenario how AVIST system helps analysts to visual explore the large network logs. Seven steps are presented in this figure, and Section 7.1 gives more detailed discussions for each step.

lutions[3], a private company that provides portfolio analysis for U.S. waterborne trade activity in the world. The dataset is about 2013 US waterborne imports from other countries, with 10,735,092 records and 10 dimensions, so total data items is 107,350,920. Among 10 dimensions, each data records features the US port code and foreign port code, which illustrates the geospatial information in each record.

We first preprocess the original dataset, and obtain the compressed data with 266MB. To facilitate analysts to explore the geospatial data, we incorporate a virtual globe view in AVIST, and its implementation follows the data dependency graph design. In this case study, we emphasis the hypothesis generation and verification. And figure 1 shows a usage scenario how analysts explore the big spatial-temporal international trades to gain insights.

Firstly, analysts specify the time window (86400 seconds or one day) before playing animation. The dynamic view shows the overview of the world trades in subgraph 1, which reveals that the international trades are clearly separated into four quarters. The second and third quarters has more transactions than others. Analysts hypothesize that the international trades may also have some spatial patterns. They highlight two US ports: Los Angeles (LA) and New York (NY). They suppose that LA has more trades with pacific countries while NY is more related with Western European because the waterborne trade may be more sensitive with geolocations. Subgraph 2 shows one snapshot of records with this two ports in virtual globe view. To verify it, analysts create two highlight filters, focusing the records from ShangHai (SH) to LA and NY. They play animation and generate the dynamic view in subgraph 3. They find that the trades from SH to LA is roughly twice of trades to NY, which supports their hypothesis. The international trades also character country's economy. Analyst generate four filters to compare the economy of China and Vietnam. They choose ShangHai port and Ho Chi Minh port for representing these two countries, and target the trades related with furniture and electronics, as shown in subgraph 4. They are more interested in the money value rather than the number of trades records, so they click the value in the listbox and play animation. Subgraph 5 is the dynamic view with four highlighted line charts, which shows that two kinds of trades are more balanced in China economy and Vietnam more relies on furniture in its exports.

[3]https://www.piers.com/

8 Discussion And Limitations

To facilitate exploratory visual analysis big data, we argue VA system should support user's drill down query. Based on this guideline, we have summarized previous techniques considering both system performance and user interactions. We emphasize the *in-situ* concept in visual analytics field to support real-time querying big data, and we present a GPU based in-situ visualization architecture, to utilize powerful GPU computing and rendering resources to achieve this. A number of human interaction techniques are also considered to help user to quickly explore big data. And we incorporate these techniques into a data dependency graph based on the GPU in-situ architecture design. As a proof of concept, we present AVIST system and show two cases to demonstrate our ideas.

There are, however, some limitations in our current system. We design to store all the raw data in the GPU memory, which limits the data size by the GPU memory capability. Even we suggest to compress the data into its binary format, the current GPU memory could only support no more than 10 million data items. Our design fails to handle big data, when the number of data scales to billion and trillion. Besides large volume, big data also features in large variety and velocity [27]. However, our current design has no good solution to cover these two areas. For example, AVIST cannot handle more than 600,000 data items at once to afford real time performance.

AVIST system only targets for exploratory visual analysis of big time-series and multidimensional datasets. By slicing such big data from time dimension and filtering from other dimensions, AVIST supports the drill-down query. In real world, big data are much more complex. FaceBook has billions of friends relationships and Amazon has tons of users, reviews, goods. Our current design and implementation cannot scale to these datasets.

9 Conclusions And Future Work

In this paper, we highlight the *in-situ* processing and visualizing in visual analysis field to real time explore big time-series and multidimensional data. We propose a GPU based in-situ visualization archi-

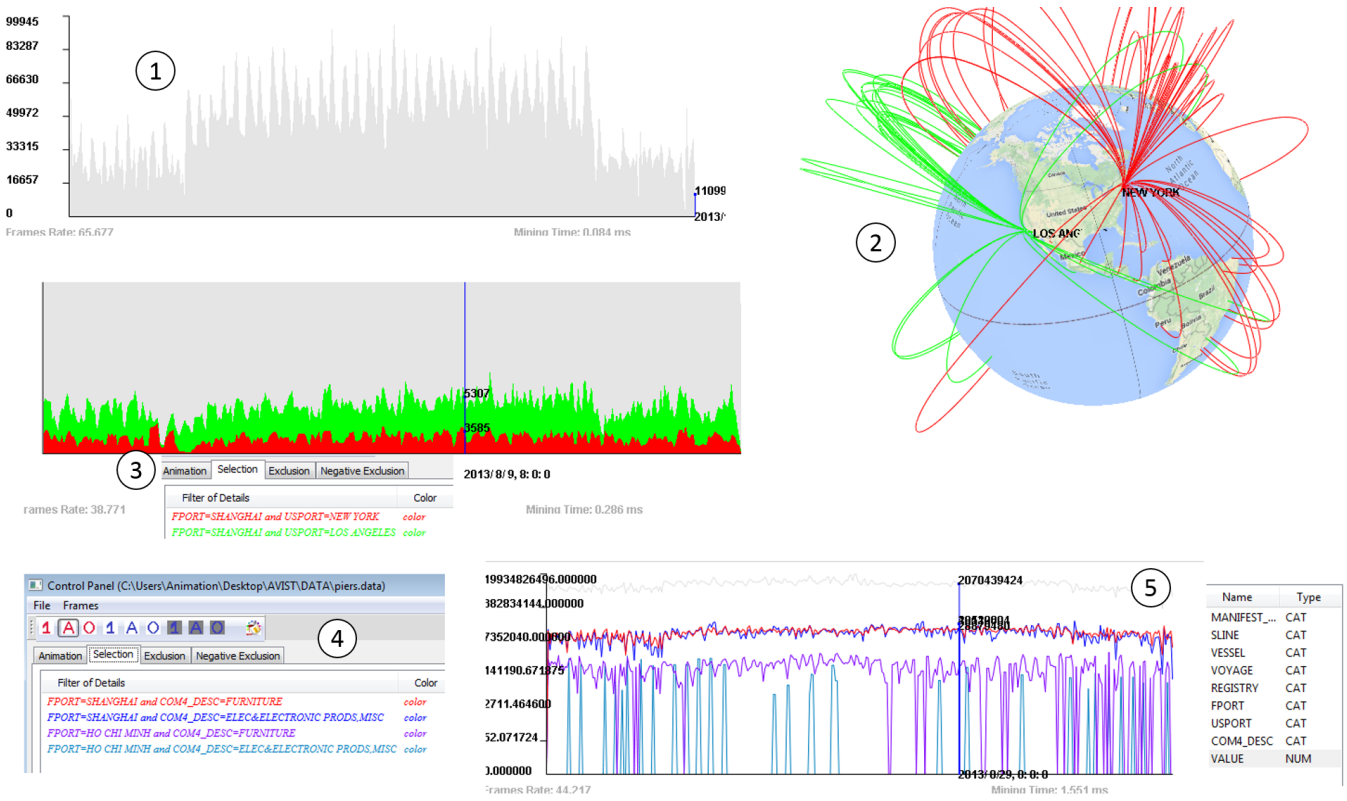

Fig. 8. This figure shows the usage scenario that analysts use AVIST system to visual explore big international trade transactions. The detailed discussions are presented in Section 7.2.

tecture to fully utilize the GPU computing and rendering capability. We implement the AVIST system as a proof of concept, and carefully design a data dependency graph in the system to incorporate several human interaction techniques. We have shown the effectiveness of AVIST system through two usage cases. Finally, we have shown the performance analysis of our system and discussed limitations.

There are several directions to continue our research. First, we want to investigate new data lossless compression techniques to feed more data on the GPU memory. Second, more data views and functionalities need be implemented to support better exploratory visual analysis. Finally, comprehensive user studies need be conducted, especially compared with the existing VA softwares such as *Tableau*[4], *Spotfire*[3], *QlikView*[2] and so on to investigate the future of VA systems.

REFERENCES

[1] Jmp software http://www.jmp.com/.

[2] Qlikview software http://www.qlik.com/.

[3] Spotfire software http://spotfire.tibco.com/.

[4] Tableau software http://www.tableau.com/.

[5] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: Queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 29–42, New York, NY, USA, 2013. ACM.

[6] L. Battle, M. Stonebraker, and R. Chang. Dynamic reduction of query result sets for interactive visualizaton. In *Big Data, 2013 IEEE International Conference on*, pages 1–8. IEEE, 2013.

[7] P. Boncz, M. Zukowski, and N. Nes. Monetdb/x100: Hyper-pipelining query execution. In *In CIDR*, 2005.

[8] Y. Chen, S. Alspaugh, and R. Katz. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *Proc. VLDB Endow.*, 5(12):1802–1813, Aug. 2012.

[9] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[10] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 2007.

[11] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1539–1148, 2008.

[12] F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees. The sap hana database–an architecture overview. *IEEE Data Eng. Bull.*, 35(1):28–33, 2012.

[13] J. D. Fekete. Visual analytics infrastructures: From data management to exploration. *Computer*, 46(7):22–29, 2013.

[14] J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, pages 117–, Washington, DC, USA, 2002. IEEE Computer Society.

[15] D. Fisher, S. M. Drucker, and A. C. König. Exploratory visualization involving incremental, approximate database queries and uncertainty. *IEEE Computer Graphics and Applications*, 32(4):55–62, 2012.

[16] D. Fisher, I. Popov, S. Drucker, and m. schraefel. Trust me, i'm partially right: Incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1673–1682, New York, NY, USA, 2012. ACM.

[17] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu. Starfish: A self-tuning system for big data analytics. In *CIDR*, volume 11, pages 261–272, 2011.

[18] J.-F. Im, F. G. Villegas, and M. McGuffln. Visreduce: Fast and responsive incremental information visualization of large datasets. In *Big Data, 2013 IEEE International Conference on*, pages 25–32. IEEE, 2013.

[19] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: A visualization-oriented time series data aggregation. *Proceedings of the VLDB Endowment*, 7(10):797–808, 2014.

[20] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. C. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg, and D. J. Abadi.

H-store: A high-performance, distributed main memory transaction processing system. *Proc. VLDB Endow.*, 1(2):1496–1499, Aug. 2008.

[21] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013.

[22] Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. *Computer Graphics Forum (Proc. EuroVis)*, 32, 2013.

[23] K.-L. Ma, C. Wang, H. Yu, and A. Tikhonova. In-situ processing and visualization for ultrascale simulations. In *Journal of Physics: Conference Series*, volume 78, page 012043. IOP Publishing, 2007.

[24] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment*, 3(1-2):330–339, 2010.

[25] K. Morton, M. Balazinska, D. Grossman, and J. Mackinlay. Support the data enthusiast: Challenges for next-generation data-analysis systems. *Proceedings of the VLDB Endowment*, 7(6), 2014.

[26] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV'07. Fifth International Conference on*, pages 61–71. IEEE, 2007.

[27] M. Schroeck, R. Shockley, J. Smart, D. Romero-Morales, and P. Tufano. Analytics: The real-world use of big data. *IBM Institute for Business Valueexecutive report, IBM Institute for Business Value*, 2012.

[28] C. Stolper, A. Perer, and D. Gotz. Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics, 2014.

[29] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):52–65, 2002.

[30] J. J. Thomas and K. A. Cook. *Illuminating the path: The research and development agenda for visual analytics*, volume 54. IEEE, 2005.

[31] C. Weaver. Cross-filtered views for multidimensional visual analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 16(2):192–204, 2010.

[32] R. Wesley, M. Eldridge, and P. T. Terlecki. An analytic data engine for visualization in tableau. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 1185–1194, New York, NY, USA, 2011. ACM.

[33] P. C. Wong, H.-W. Shen, C. R. Johnson, C. Chen, and R. B. Ross. The top 10 challenges in extreme-scale visual analytics. *IEEE computer graphics and applications*, 32(4):63, 2012.

[34] L. Zhang, A. Stoffel, M. Behrisch, S. Mittelstädt, T. Schreck, R. Pompl, S. Weber, H. Last, and D. Keim. Visual analytics for the big data era A comparative review of state-of-the-art commercial systems. In *IEEE Conference on Visual Analytics Science and Technology 2012, VAST 2012 - Proceedings*, pages 173–182, 2012.

[35] J. Zhao, C. Collins, F. Chevalier, and R. Balakrishnan. Interactive exploration of implicit and explicit relations in faceted datasets. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2080–2089, 2013.

[36] M. Zinsmaier, U. Brandes, O. Deussen, and H. Strobelt. Interactive level-of-detail rendering of large graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2486–2495, 2012.