Evaluating Data Interaction: Workloads, Metrics, and Guidelines

ACM SIGMOD 2018 Tutorial – Extended Version

Lilong Jiang
Twitter, Inc
San Francisco
jianglil@cse.ohio-state.edu

Protiva Rahman
Computer Science & Eng.
The Ohio State University
rahmanp@cse.ohio-state.edu

Arnab Nandi
Computer Science & Eng.
The Ohio State University
arnab@cse.ohio-state.edu

ABSTRACT

Highly interactive query interfaces have become a popular tool for ad-hoc data analysis and exploration. Compared with traditional systems that are optimized for throughput or batched performance, ad-hoc exploration systems focus more on user-centric interactivity, which poses a new class of performance challenges to the backend. Further, with the advent of new interaction devices (e.g., touch, gesture) and different query interface paradigms (e.g., sliders, maps), maintaining interactive performance becomes even more challenging. Thus, when building and evaluating interactive data systems, there is a clear need to articulate the evaluation space.

In this paper, we describe unique characteristics of interactive workloads for a variety of user input devices and query interfaces. Based on a survey of literature in data interaction, we catalog popular metrics for evaluating such systems, highlight their deficiencies, and propose complementary metrics that allow us to provide a complete picture of interactivity. We motivate the need for *behavior-driven* optimizations of these interfaces and demonstrate how to analyze and employ user behavior for system enhancements through three scenarios that cover multiple device and interface combinations. Our case studies can inspire guidelines to help system designers design better interactive data systems, and can serve as a benchmark for evaluating systems that use these interfaces.

1. INTRODUCTION

Interactive data systems that allow ad-hoc, iterative analysis with instant feedback are becoming increasingly common [3, 17, 34, 38, 60, 91]. This trend is evidenced by open-source tools such as crossfiltering [3] which allows users to explore large (approximately one million records) multidimensional datasets interactively in a coordinated view, running in a standard web browser. Other examples include LiveRAC [91] which allows side-by-side visual comparison at different zoom levels and Google Charts [8] which allows users to build different types of interactive charts. Each such tool or query interface generates a unique workload (Section 2), which needs to be considered when evaluating and optimizing it. In fact, real world systems at times employ a combination of these workloads, further complicating the optimization process. For example, Tableau [17] allows its users to compose queries via text box (keyword search), slider (query by sliding), and map (query by zooming and dragging).

This increase in interactive systems has gone hand-in-hand with increase in touch-based (e.g., iPad, Microsoft Surface), and gesture-driven devices (e.g., Kinect, HoloLens, Leap Motion). The shipment of touch-screen displays is estimated to be 2.8 billion in 2016 [6]. Further, the rise of virtual and augmented reality has popularized gestural querying, which is being used in wider applications including automobile [5, 111] and health care [19, 100] industries. These rates of adoption suggest that touch and gestures will likely be the primary mode of data interaction in the near future. In fact, recent work by multiple members of the community have shown that touch devices can be more usable and intuitive than mouse for data interaction, and that gestural devices have performance that can be considered comparable to mouse-driven interaction [64, 85, 95, 132]. Tableau Vizable [18] and Microsoft Power BI [15] are two such examples of touch-based data analytics tools.

This trend towards non-keyboard based interactive data analysis motivates the need for a deeper study into the workloads generated by these systems along with methods for evaluating them. Prior work on non-keyboard systems has focused on usability studies, while those on interactive systems has focused on performance studies (Table 2). Here, we look at the intersection of these two. This requires a detailed analysis of the user's interactive querying behavior, taking into consideration the unique characteristics of different device-interface combinations (Section 3). While Eichmann et al. [46, 47] provide some metrics and workloads for interactive data exploration systems, they do not account for user behavior and non-keyboard devices. In contrast, we not only provide workloads from user studies on traditional mouse and keyboard, but also from non-keyboard devices such as leap motion and touch screens. Further, we demonstrate the employment of *behavior-driven* optimiza-

Table 1: Interactive Workloads Summary. The workloads span different devices, interfaces, interaction techniques, and queries.

name	device	query interface	interaction techniques	trace	query
inertial scrolling (Section 4)	touch(trackpad)	scroll	browsing	{timestamp, scrollTop, scrollNum, delta}	`select`, `join`
crossfiltering (Section 5)	mouse, touch(iPad), gesture(leap motion)	slider	linking & brushing	{timestamp, minVal, maxVal, sliderIdx}	`count` `aggregation`
filter and map (Section 6)	mouse	textbox, slider, checkbox, map	filtering & navigating	{timestamp, tabURL, requestId, resourceType, type, status}	`select`, `join`

tions for these systems and show results of evaluating existing databases using these workloads and optimizations.

This paper is organized as follows: In Section 2, we talk about characteristics of data interaction for different devices and interfaces. Section 3 summarizes metrics for query interfaces in current literature and new ones we propose. We then walk through three case studies of evaluating interactive systems: Inertial scrolling in Section 4, Crossfiltering in Section 5 and composite systems (specifically filter and map) in Section 6. In each case study, we describe the experimental setup, analysis of user behavior and suggest *behavior-driven* optimizations. Finally, we conclude and describe future directions in Section 8.

Contributions: The goal of this paper is to highlight the unique characteristics of evaluating interactive systems, and demonstrate methods to address them through case studies. The results from the case studies can also be used to inform optimizations for system designers who are implementing similar interactive query stacks. This paper makes the following contributions:

- We describe salient features of interactive workloads with respect to multiple devices-interface combinations.
- We catalog and summarize data interaction metrics being used currrently in relevant literature, and also propose some new metrics.
- We demonstrate and provide guidelines for using *behavior-driven* optimizations and their impact on performance improvements through three case studies.

2. SALIENT FEATURES OF INTERACTIVE DATA SYSTEMS

Workloads generated by interactive systems have salient characteristics that distinguish them from traditional database workloads, which we summarize here.

2.1 Devices and Interfaces

Different input devices have different sensing rates for user input, which directly affect the *query issuing frequency*, i.e., number of queries per second. The query workload generated by each interface is also different. For example, one zoom action on a map interface triggers multiple predicate changes (longitude and latitude) in the `WHERE` clause. Mouse or touch-based sliders tend to trigger more intensive workloads (number of queries per second) than text input since typing typically takes more time than sliding.

2.2 Continuous Actions

For touch and gesture devices [95] used in conjunction with continuous manipulation query interfaces (e.g., slider, linking and brushing) [45], the user's query specification has to be treated as a continuous process, with each change triggering a query. In direct manipulation systems, the whole process needs to remain interactive, motivating the need for a strict latency constraint. The fluid nature of this interaction generates heavy query workloads. Consider for example the case of *crossfiltering*, where the user moves the slider continuously to filter the dataset. If the user interface detects events every 20ms, then every second about 50 queries are issued to the backend from a single user.

2.3 Ambiguous Query Intent

When a user interacts with the mouse and touch devices, they are manipulating physical objects. The presence of friction and force make these interactions more accurate. However, on gestural devices, due to the absence of friction the interaction process is highly variable and sensitive [64]. The user's difficulty in holding the cursor steady at a specific point is compounded with the sensor's ability to detect minor hand movements. These effects easily trigger unintended, noisy, and repeated queries more frequently as compared to mouse and touch. Figure 5 shows the trace for mouse and gesture devices. In designing workloads, we need to account for the system's ability to deal with unintended queries.

2.4 Dependent Queries

Interactive systems have a concept of sessions, where the user's behavior in a session is related, since they might be trying to answer a specific question. Thus, *adjacent queries*, i.e., those issued in adjacent timestamps, are generated when the user performs continuous gestures. Adjacent queries usually return related or similar results since there is usually only a difference of one condition (e.g., `where` clause predicate) between them. This dependence can be used for performance optimizations.

2.5 Confounding Factors and Biases

Additionally, most interactive systems require user studies or analysis of user behavior. In addition to selecting the right metrics, system designers need to consider bias and externalities when conducting user studies. There is a wide body of literature in the HCI community [24, 27, 49] which discusses these factors, which include:

1. *Learning*: In a within-subject study, each user has to complete the same task on the system being tested as well as on the control. This leads to the user doing slightly better on the second system, since they are

Table 2: Metrics for Data Interaction in Current Literature

type	metric	explanation & references
query interface	usability	Proxied by query specification or task completion time [23,26,28,44,61,62,63,66,87,95,103,109,125, 126,133], number of iterations, navigation cost [29,68,69,84], miss times [64], number and uniqueness of insights [48,101,109,121], accuracy [99,108], ability to complete tasks [28,36,57,73,87,109,125,126]
	learnability	ability to learn user actions with instructions: [28, 95, 101]
	discoverability	ability to discover user actions without instructions: [26, 64, 90, 95]
	user feedback	survey questions: [36, 57, 61, 65, 66, 68, 87, 95, 101, 121], case study [33, 51, 52, 77, 91, 97, 105, 119, 123, 127, 128, 131], design study [104, 108],suggestions [28, 35, 36, 63, 74, 92, 99, 103, 108, 109, 119, 122]
	behavior analysis	sequences of mouse press-drag-release [63], event state sequence [81]
performance	throughput	transactions / requests / tasks per second: TPC-C, TPC-E [20], [37]
	latency	execution time of the query or frame rate per second: [37,44,50,58,64,65,68,74,86,88,89,99,108,114]
	scalability	performance with increasing data size, number of dimensions, number of machines: [42, 44, 65, 75, 89, 99, 114]
	cache hit rate	[31, 65, 112]

familiar with the task. In order to combat this, experiment designers need to randomize the order in which the user is exposed to the system.

2. *Fatigue*: Long tasks can lead to users performing poorly towards the end, due to fatigue. Hence, tasks need to be broken into small chunks, with adequate breaks in between.

3. *Carry-over effects*: In within-subject studies, this refers to the user's performance on the second task being affected due to completing the first task. This differs from learning in that their performance in the second task could diminish. For example, if both the control and the experimental system are new to the user, they may perform poorly on the second by confusing functionalities with the first. Randomization can help account for this. However, if the effects are asymmetric, i.e., one user shows improvement while the other one shows deterioration, it makes it hard to draw conclusions.

3. METRICS

Table 2 summarizes metrics currently used for evaluating interactive systems. Some common qualitative metrics for *query interfaces* include:

1. Usability: Captures ease of use of the interface
2. Learnability: Captures if users are quickly able to learn how to use the system, after being taught
3. Discoverability: Captures if users are able to learn how to use the system without any instructions.

Common performance metrics include latency and scalability [64, 65]. Metrics for visualization systems are summarized in [80, 94]. Even though both, query interface and performance, are critical for a system's evaluation, we find that only few systems [44,64,65,68,74,99,108] evaluate both facets.

Latency Constraint Violations: A key metric which we find missing in prior work is perceived latency violations. Latency is a common metric used for measuring user wait times. However, in an interactive system, many queries are not issued in isolation: it can be dependent on the result of another query, causing cascading failures. Alternatively any latency introduced can break a user's "flow" – the user's experience diminishes if they perceive any latency at all (bounds have been discussed by Liu and Heer [88]). Hence, the goal for all systems should be zero perceivable latency. In order to capture this stricter notion, we introduce *latency constraint violation*. Specifically, this metric counts the number of times the zero latency rule is violated i.e. the user perceives a delay. We describe this metric in the context of two of our case studies.

There are a variety of metrics employed by systems based on their use-case, but no set of standard metrics are adequate for capturing the whole picture, making it difficult to compare two systems. Given the variety of interfaces and their uniqueness outlined in Section 2, this is understandable. However, this process becomes easier if certain principles, such as the following, are practiced when evaluating new systems.

1. As mentioned earlier, system designer should take *behavior-driven* optimizations into consideration, leveraging the user's session characteristics when designing and evaluating interactive systems.
2. Metrics should maximize the coverage of query types (e.g., `select`, `join`, `aggregation`) and interaction techniques [71, 106, 120, 130] (e.g., filtering, linking & brushing), since each of these generate unique workloads.
3. User study tasks should simulate *real-world* use cases on *real* datasets, because interactive systems are used everyday as opposed to data warehouses used for business transactions.
4. When studying user behavior, some studies recommend a minimum of 10 users [49, 82]; however, this number strongly depends on the nature of tasks and variability of the interaction itself.
5. Evaluations should cover a variety of workloads, e.g., different scenarios, data distributions, data sizes, etc.

We demonstrate these principles (except the last one which can be controlled synthetically) by walking through three case studies. The behaviors studied and metrics used in the case studies are summarized in Table 3.

4. INERTIAL SCROLLING

Table 3: Metrics Summary

interface	behavior	performance
inertial scrolling (Section 4)	scrolling speed	latency constraint violation
	no. of backscrolls	latency
crossfiltering (Section 5)	sliding behavior	query issuing frequency
	querying behavior	latency
		latency constraint violation
filter and map (Section 6)	query interface	exploration time
	zooming	
	dragging	data request time
	filter conditions	

Scrolling is a common way to browse information when the result set does not fit on the screen [61, 109]. Inertial or momentum scrolling [93] is a feature that helps users scroll smoothly and is widely adopted in touch based devices e.g. smartphones, trackpads, etc. Its salient feature is that there is an acceleration when the user scrolls. Further, when the user stops scrolling, the screen stops gradually as opposed to immediately as in traditional scrolling.

In scrolling interfaces, large result sets are impossible to load at once since they do not fit in memory. Moreover, the user cannot digest the entire result set at once. Different loading strategies are used to address this, such as lazy loading implemented using `LIMIT` and `OFFSET`. We conduct a user study to collect scrolling traces under ideal conditions i.e. without loading latency, and then use these traces to optimize loading strategies.

Dataset: We selected the top rated 4000 tuples, the same number as [109], from the IMDB [9,13] movie dataset, with 6 attributes (poster, title, director, genre, plot, rating). All tuples are preloaded to the browser to avoid loading latency.

Task & Users: 15 users were asked to *skim* all 4000 tuples and select interesting movies by scrolling through a MacBook [22] trackpad. Each user is trained on scrolling on the touchpad before the task.

Trace & Query: For each scroll and wheel event, we store the current timestamp, scrollTop [16] which is the number of pixel scrolled upward, number of tuples scrolled, and delta [4] which is the accelerated scroll amount. The query can be a simple `select` query (Q1), which selects 100 tuples every time,

```
SELECT poster, title || '(' || year || ')',
director, genre, plot, rating
FROM imdb
LIMIT 100 OFFSET 100
```

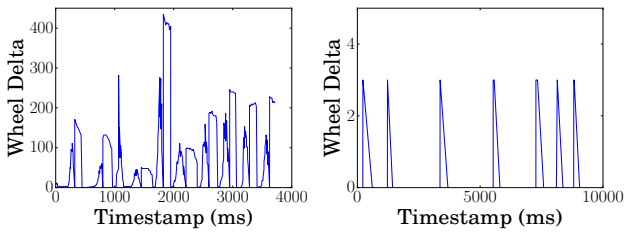
or a complex streaming `join` query (Q2) [67, 115] when the information comes from different streaming sources.

```
SELECT poster, title || '(' || year || ')',
director, genre, plot, rating
FROM (
(SELECT id, rating
FROM imdbrating
LIMIT 100 OFFSET 100) tmp
INNER JOIN movie ON
tmp.id = movie.id
)
```

In the workload, we vary the `limit` and `offset` numbers for both queries.

4.1 Behavior Analyses

We study the scrolling speed and number of back scrolls to capture the user's difficulty in selecting targets.



(a) with inertia. (b) without inertia.

Figure 1: Scrolling with / without inertia: The wheel delta indicates the distance scrolled. As can be seen from the plot, the user scrolls much faster with inertial scrolling (y-axis scale: 400 vs 4), rendering lazy loading ineffective.

Scrolling Speed: Figure 1 shows the wheel delta i.e. distance scrolled, against the timestamp from part of one user's trace. The figure indicates that the user scrolls larger distances (y-axis scale: 400 vs 4) with inertial scrolling than regular scrolling. Hence, techniques such as lazy loading [32, 55] i.e. fetching more items once the user reaches the bottom of the results do not work. The user often reaches the end of the page before items are loaded, which increases wait time and reduces usability.

No. of Backscrolls: We also observed that the increased momentum often causes users to scroll past a movie that they want to select, requiring them to scroll back to select it (Figure 2). These findings verify the needs of designing interfaces that reduce information loss, as discussed in previous research [61, 109].

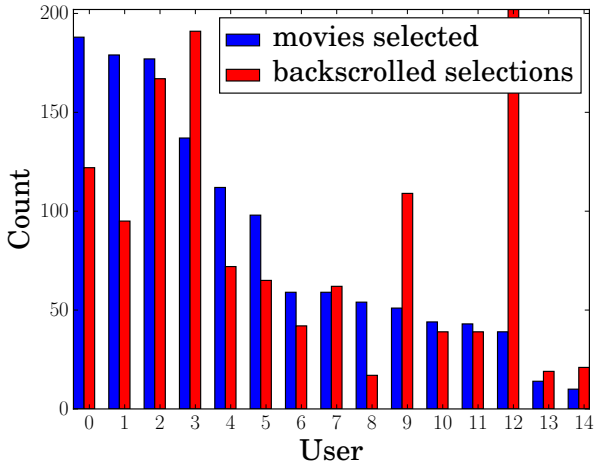


Figure 2: Comparison of number of selected movies and number of movies selected with backscrolls. In some cases, the number of backscrolls is larger than the number of selected movies, indicating that the user had to scroll back and forth multiple times to select a movie. This shows the need for additional usability optimizations with inertial scroll.

4.2 Evaluation

In this section, we evaluate query workloads generated from the user study over PostgreSQL. For each query, the reported query execution time is the maximum of 10 runs. To decide on number of tuples to fetch, we compared {12, 30, 58, 80} tuples. These values correspond respectively

to lower bound of maximum, upper bound of average, median of maximum, and mean of maximum scrolling speed (measured in no. of tuples) of 15 users. We compare two alternative prefetching strategies to lazy loading:

1. *Event fetch:* For every scroll event, the system checks whether there are enough prefetched items in cache. If not, then more items are prefetched. Since a scroll event is triggered every 15–20 ms, this method adds heavy computation burden on the browser. For our experiments, we set cache limit to product of tuples to fetch and query execution time.

2. *Timer fetch:* Items are prefetched at regular time intervals e.g. 20 items every second. We set the fetching interval to 1 second and fetch a fixed number of tuples.

Latency: An obvious metric for comparing these strategies is latency in loading tuples, shown in Figure 3. As can be seen *event fetch* is insensitive to the number of tuples fetched and keeps at a reasonable latency $\sim$ 80 ms. In *timer fetch*, when number of tuples fetched is low, faster users can wait up to $\sim$ 60 seconds. However, it decreases with increase in of tuples and achieves zero latency when the tuples fetched equals the median of max scroll speed.

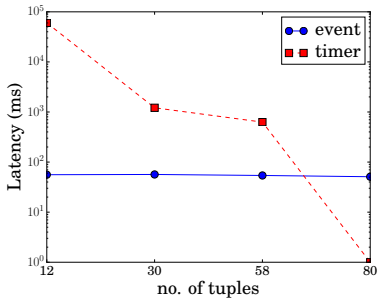


Figure 3: Average latency of 15 users over different number of tuples fetched. *Event fetch* is insensitive to the number of tuples. *Timer fetch* decreases linearly reaching zero latency at median of max scrolling speed.

Latency Constraint Violation: For inertial scroll, a latency constraint violation happens if the number of tuples scrolled is greater than the cumulative number of tuples cached so far, since the user will have to wait for tuples to load. Table 4 shows the number of users (out of 15) who observed latency constraint violation. The total number of constraint violations are also shown for each of the four values of tuple numbers. For *timer fetch*, violations only occur in few users' traces. When we increase the number of tuples fetched, the number of violations rapidly decreases. However, *event fetch* is less sensitive to the number of tuples fetched since it only fetches more tuples when the number of currently cached tuples is less than the cache limit. Because of the acceleration, at some points, the number of tuples scrolled may be greater than the number of cached tuples.

5. CROSSFILTERING

Crossfiltering (or brushing-and-linking [71]) is a popular visualization concept which allows users to issue multi-dimensional filters to a subset and explore the dataset in a coordinated view arrangement [3, 64, 86, 89, 118]. Figure 4 shows the UI for our experiments, where each histogram corresponds to one attribute and is controlled by a range slider.

Table 4: Latency Constraint Violations for Event & Timer Fetch: For timer fetch, the number of violations decreases with number of tuples cached, and it performs better than event fetch in general.

# tuples fetched	12	30	58	80
# users (event)	15	15	15	14
# users (timer)	3	1	1	0
# no. of violations(event)	2203	840	457	167
# no. of violations (timer)	767	2	1	0

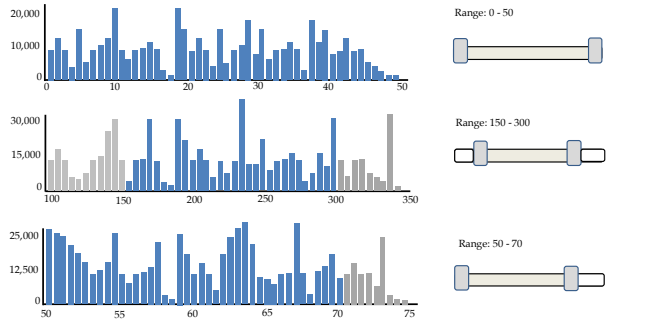


Figure 4: Crossfiltering Interface. Each histogram corresponds to one attribute and range slider. The blue indicates the filtered range for each attribute. Histograms for other attributes are updated synchronously while the user is manipulating one slider.

Dataset: We used a 3D road network dataset from the UCI ML repository [21, 70], which has 3 attributes (`longitude`, `latitude` and `height`) and 434,874 tuples.
Task & Users: We recruited 30 users (10 per device) and asked them to specify range queries by moving the handle to a specific position with mouse, iPad or Leap Motion. The crossfiltering library [3] was used, which can support extremely fast (<30ms) interactions for up to one million records. Hence, users did not perceive any query latency.
Trace & Query: For each sliding event, we collect the current timestamp, value range, and slider index. At each timestamp, two queries are issued concurrently. Each query is used to calculate a histogram. One query example is shown below:

```
SELECT ROUND((y - 56.582) / ((57.774 - 56.582) / 20)),
COUNT(*)
FROM dataroad WHERE x >= 8.146 AND x <= 11.2616367163
AND y >= 56.582 AND y <= 57.774
AND z >= -8.608 AND z <= 137.361
GROUP BY ROUND((y - 56.582) / ((57.774 - 56.582) / 20))
ORDER BY ROUND((y - 56.582) / ((57.774 - 56.582) / 20))
```

Configuration: We run queries on PostgreSQL [14] and MemSQL [11] on a Linux machine (Intel Core i5-4590 CPU @ 3.30GHz $\times$ 4, 15.6 GiB Memory). PostgreSQL is a popular disk-based database while MemSQL is an in-memory database. We choose these two databases since we wanted to compare the interactive performance for disk-based and in-memory databases. In Python, the global interpreter lock ensures that only one thread runs in the interpreter at once [7]. In order to issue multiple queries at the same time, we fork and execute multiple Python processes at the same time, where each process has an independent database

connection.

5.1 Behavior Analyses

We describe the sliding behavior, i.e., the manner in which the user interacts with the sliders, and how this affects their querying behavior, i.e., the manner in which queries are issued.

Sliding Behavior: Figure 5 shows representative traces on two different devices from the same user. As seen on the trace, Leap Motion triggers unintended gestures, resulting in heavy workloads from noisy queries.

Querying Behavior: Since crossfiltering uses sliders as the query interface, queries are issued one after another as the handle is moved continuously. When the user interface frame rate per second is 20ms, every second about 50 queries are issued. In coordinated views, such as crossfiltering, about $50(n-1)$ queries are issued, where n is the number of dimensions, leading to heavy workloads. We propose two optimization algorithms to reduce this, which can be applied to any interface that has high query frequency.

1. *Skip:* In crossfiltering, no dependency exists between adjacent queries, since each represents a separate range query i.e. unlike in inertial scroll where the user browses serially, a user does not necessarily look at ranges serially. Thus, queries can be *skipped* to improve performance. If the query execution time is high, the user might have issued a second query before the first one completes (Figure 8). Further, in exploratory data analysis, the user may abandon issued queries if initial findings do not match the hypothesis being tested. In this case, previously issued queries for that hypothesis should also be abandoned i.e. the current query run by the database may already have been skipped by the user. Based on this, a natural optimization would be to skip previous queries once a new one has been issued.

2. *KL:* Most adjacent queries have same or similar results, since users make iterative changes. Hence, an alternative optimization would be to only execute queries whose results differ to a certain extent from the previous query. Hash-based methods [56], sampling-based methods [39, 53, 98], wavelets-based methods [116] etc. can be used for fast approximation of histograms i.e. result sets, without running the query. We use the Kullback-Leibler (KL) [79] divergence to measure the difference between two histograms, before queries are sent to the datasbase. If KL = 0, then the two histograms are the same. We approximate KL by quantizing and comparing histograms $\mathcal{T}$ and $\mathcal{T}'$ at each of the n bins: $KL(\mathcal{T}, \mathcal{T}') = \sum_{x=0}^{n} \mathcal{T}(\frac{x}{n}) \times ln(\frac{\mathcal{T}(\frac{x}{n})}{\mathcal{T}'(\frac{x}{n})})$. In [124], the authors present initial work on measuring human's perception error of with KL, which can be used to determine the KL threshold.

5.2 Evaluation

We evaluate query issuing frequency and latency over three variable factors: databases (PostgreSQL, MemSQL), devices (mouse, touch, leap motion), and optimization methods (*raw*, KL>0, KL>0.2, *skip*), where *raw* means running all queries.

Query Issuing Frequency: To demonstrate the high frame rate in interactive systems, we plot the frequency histogram of query issuing intervals from one trace for each device. Figure 6 provides us with the following insights:

1. The number of queries issued by the leap motion is much larger than mouse and touch (y-axis scale: 2500 vs 120), which verifies the instability and sensitivity of leap motion.

2. If we only issue queries when there is a difference in result sets (KL>0), we can drastically reduce the number of queries issued.

3. Compared to the leap motion whose frequencies concentrate on 20ms - 25ms, the shape of the bell is broader for mouse and touch. This indicates that leap motion has a stricter latency requirement than mouse and touch.

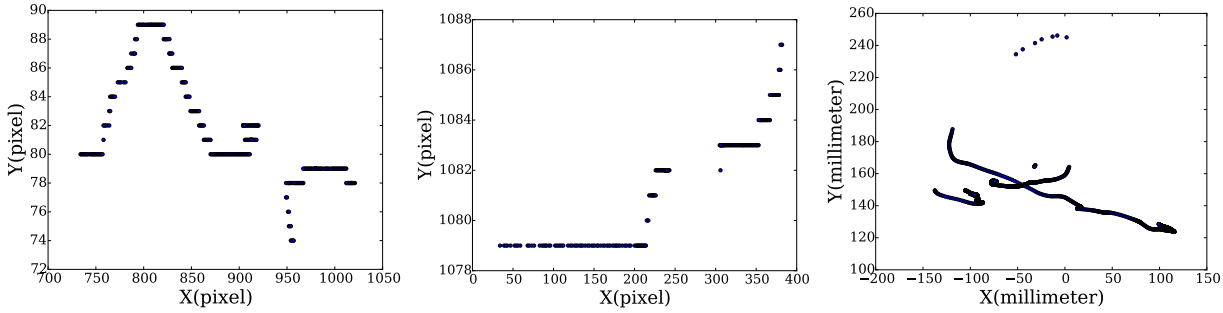
Latency: We plotted the latency of each user under each condition and device, and show a representative one in Figure 7. The trends do not vary much among the different devices, hence we only show plots for the mouse. It is clear from the figure that MemSQL can maintain a latency of 10∼50ms for all three algorithms. For KL>0, it can maintain an interactive performance ∼10ms. The *skip* strategy doesn't work better than the *raw* since the latencies of most queries are below the query issuing intervals. On the other hand, PostgreSQL's latencies are beyond 10secs for *raw* and KL>0. When we skip queries and run queries with KL>0.2, it can keep a latency 100∼1000ms (< 1sec).

Latency Constraint Violation: For crossfiltering, a latency constraint is violated if the user issues a second query before the results of the first are returned. This is shown in Figure 8, where Q1, Q2, Q3 all violate the constraint since their execution time is beyond the query issuing interval. Further, these violations cascade, since the delay in Q1's execution adds to the delay in Q2's execution and so on.

We evaluated the percentage of violations for the skip and KL optimizations on the three devices and two databases, shown in Figure 9. As expected, MemSQL has a lower percentage of violations than PostgreSQL. When we issue queries with KL>0, we can reduce about half of violated queries for MemSQL. For PostgreSQL, however, queries have to be issued with KL>0.2, for observable differences. In fact with KL>0.2, we achieve 30% decrease for mouse and touch and 17% decrease for leap motion for PostgreSQL. We find that the running times of the violated queries are between 150-500ms for PostgreSQL, and < 25ms on MemSQL.

6. FILTER AND MAP

Most interactive systems [15, 17] allow users to explore datasets through multiple query interface widgets (e.g., text box, slider, map, etc.), which we try to simulate in this case study. A key technique for improving performance in interactive systems is speculative prefetching. Many methods have been proposed for prefetching, such as Markov chains [78,83], heuristic methods [129], and data-driven characteristics [31]. However, they have either only been evaluated using synthetic workloads [78, 83, 129] and prototype systems [78], or they have focused on one query interface, like map [31, 129], which is easier to predicate than multiple query interfaces.



(a) Mouse (b) Touch (c) Leap Motion

Figure 5: Traces of a user specifying a range query on three devices: The leap motion presents more jitter than mouse and touch.

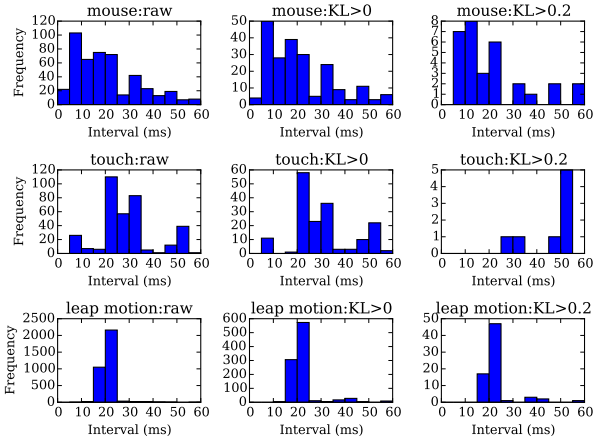


Figure 6: Frequency Histograms of Query Issuing Intervals

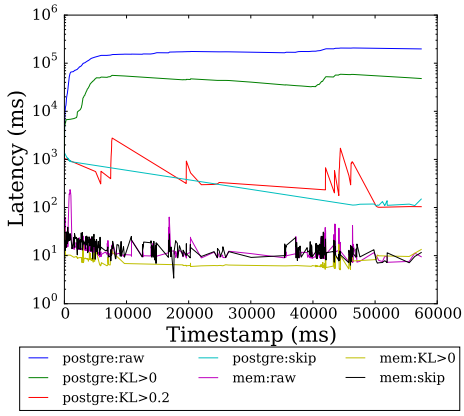


Figure 7: Latency for Mouse under Different Factors. MemSQL can maintain a latency of 10∼50ms. After some optimization (KL=0.2 or skip), PostgreSQL can maintain a latency 100∼1000ms.

In this user study, we use a popular commercial accommodation website – Airbnb [1](shown in Figure 10) as the experimental setup since it satisfies the requirement of having multiple query interface widgets and is freely accessible. Our goal in this case study is to analyze traces of the user's interaction with different widgets on *a real commercial website*, in order to inform parameters on *behavior-driven* prefetching techniques. They can also serve as a public benchmark to evaluate current speculative prefetching techniques.

Task & Users: We recruited 15 students and asked them to think of an ideal vacation and then use Airbnb to book short-term housing for it. In order to get enough traces, we asked users to spend at least 20 minutes on it.

Trace & Query: We wrote a browser extension to collect HTTP requests, tab URLs, and events. In order to get a clean requests collection, we only collected requests whose method was GET. For these requests, we collected data, image, and map requests, ignoring other requests such as status update. For each HTTP request, we further recorded its request id, timestamp before the request is made, timestamp after request is collected, resource type (e.g., image, XMLHttpRequest, etc.), and its URL. The Airbnb tab URL itself can be regarded as a query:

```
https://www.airbnb.com/s/Alabama--United-States?page=1
&source=map&airbnb_plus_only=false
&sw_lat=27.7347654065365 4&sw_lng=-91.1290339635413
&ne_lat=36.80215351286778&ne_lng=-82.0982722447913
```

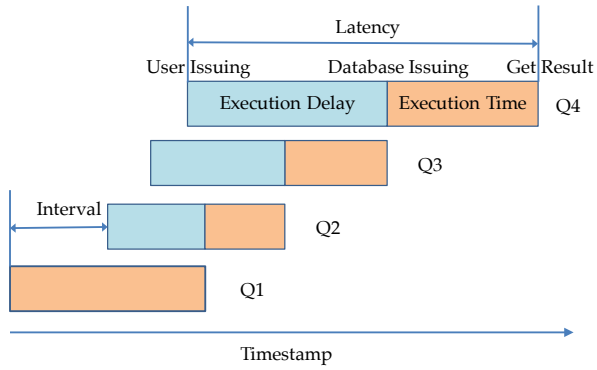


Figure 8: Latency Constraint Violation: Before Q1 finishes its execution, Q2, Q3 and Q4 are already issued by the user in the front-end. When executing Q4, there is already the execution delay caused by previous queries.

```
&search_by_map=true&zoom=6&checkin=12%2F14%2F2016
&checkout=12%2F18%2F2016&guests=3
&room_types%5B%5D=Entire%20home%2Fapt&price_min=10
&price_max=56&allow_override%5B%5D=
```

From this URL, we can get query parameters such as

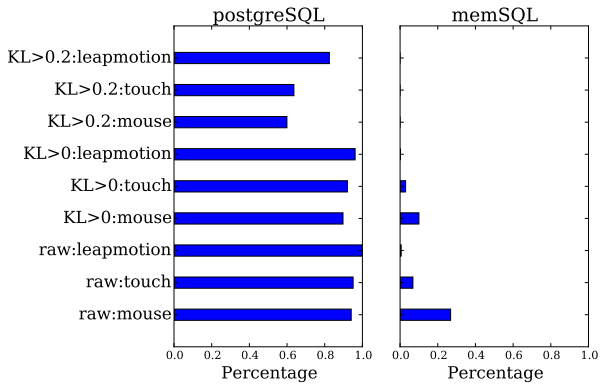


Figure 9: Percentage of Queries that Violate Latency Constraint. KL>0 is enough to improve performance in MemSQL, but KL>.2 is required for observable differences in PostgreSQL.

place, check-in time, check-out time, map, southwest longitude.

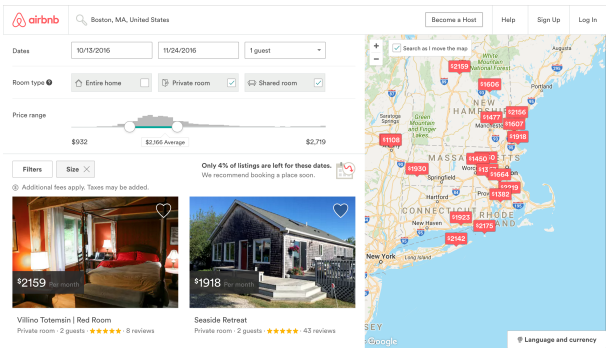


Figure 10: Airbnb Interface which incorporates different query widgets, e.g., slider, map, etc.

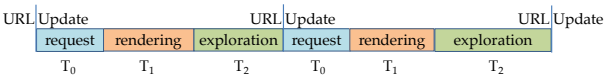


Figure 11: Exploration Process

6.1 Behavior Analyses

Exploration Process: The user' s exploration process can be modeled as shown in Figure 11. The session starts when the tab URL is updated. Then the browser waits for the data and resources of the request (request time T_0), and renders them (rendering time T_1) once they are received. Finally, the user spends some time exploring the results and decides the next query (exploration time T_2). We estimate the rendering time by listening to mutation events (e.g., DOMNodeInserted, DOMNodeRemoved, DOMSubtreeModified) [12] and recording their timestamps.

Query Interface: We report the percentage of queries issued using each interface widget in Table 5. It shows that the user prefers issuing queries with map as opposed to other interfaces. These percentages can be used for informing weights in prefetching strategies, e.g., more map tiles should be prefetched as opposed to the next range for slider.

Table 5: Percentages of Queries for Each Interface

interface	map	slider, checkbox	button	text box
percent	62.8%	29.9%	3.6%	3.6%

Zooming: Next, we analyzed the different zoom levels used by users' in their exploration, to find the most common levels for prefetching. Figure 12 shows the change of zoom levels over time, where each user is represented by a color. It shows that except for one, users zoom navigate at most, three level from their starting point, so depths greater than three can be ignored when prefetching. The majority of users explore between levels 11 and 14. Most precomputation efforts should thus concentrate on these levels.

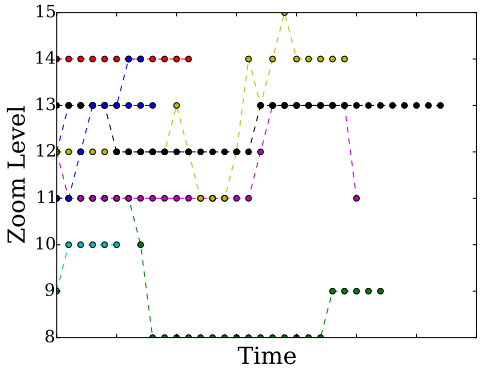
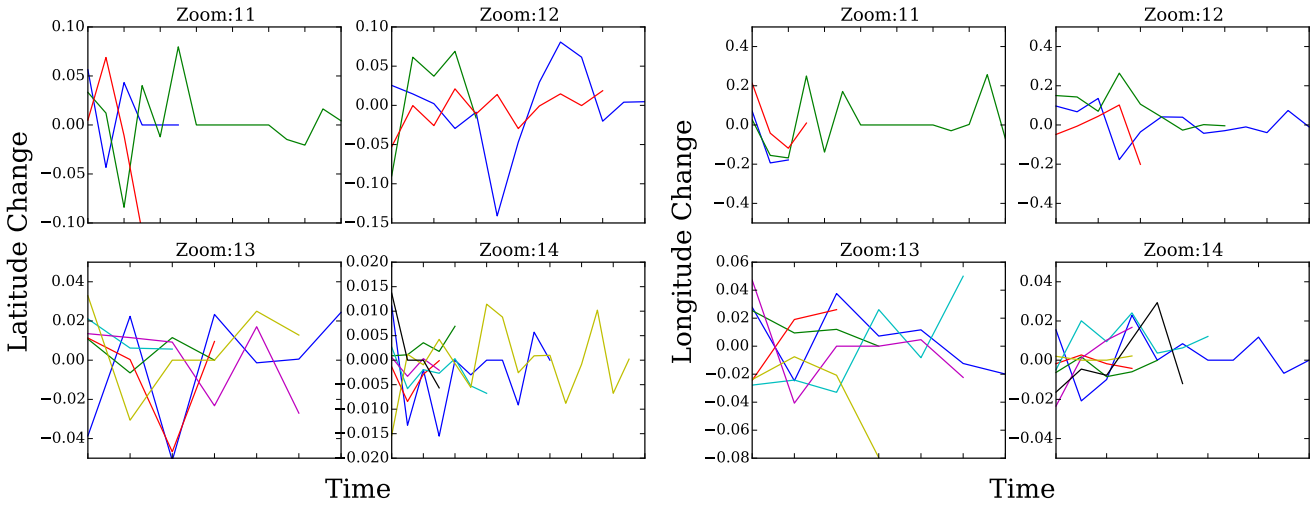


Figure 12: Change of Zoom Levels over Time for each user, represented by a color. Zoom levels concentrate between 11 and 14.

Dragging: We further analyzed map dragging behavior for a particular zoom level, focusing on levels 11–14, since these were the most common. We facet over different zoom levels since, at different levels, the same viewpoint size represents a different area size. Figure 13 shows the longitude and latitude change of the bound center (i.e., average of the four bound corners) over the time for different users. Table 6 summarizes this in the form of ranges of latitude and longitude change. As we can see, at deeper levels, users move smaller distances. These statistics can be used to determine number of adjacent tiles to prefetch as well as determine the appropriate tile size.

Table 6: Ranges for Center of Bounds

zoom	latitude	longitude
11	[-0.10, 0.07]	[-0.2, 0.2]
12	[-0.15, 0.07]	[-0.2, 0.2]
13	[-0.05, 0.03]	[-0.08, 0.05]
14	[-0.015, 0.013]	[-0.02, 0.02]



(a) Latitude

(b) Longitude

Figure 13: Change of Latitude / Longitude of the Bound Center over Time for All Users

Filtering Behavior: Figure 14 shows the cumulative distribution for number of filter conditions i.e. frequency of queries with 2 filters, 4 filters and so on. It shows that 70% of queries had four filters or less. Thus, it makes more sense to cache results with upto 4 filter predicates, which can be refined as more filters are used.

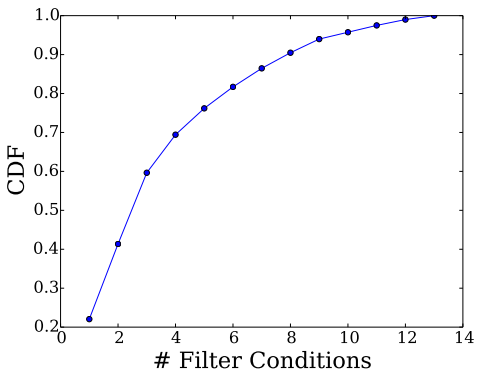


Figure 14: Cumulative Distribution Function (CDF) of Number of Filter Conditions used.

6.2 Performance Experiments

Figure 15 shows the cumulative distributions for the request and exploration times over all users. It shows that 80% of exploration time are greater than 1 second and 80% queries are completed in less than 1 second, which indicates that at least one query can be fetched in most cases. On average, the exploration time is 18.3 seconds and the fetching time is about 1.1 seconds, which indicates that about 18 adjacent queries can be fetched.

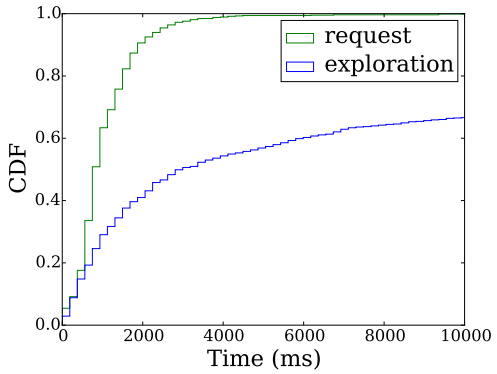


Figure 15: Cumulative Distribution Functions (CDFs) for Request and Exploration Time: The request time is much lower than the exploration time indicating that multiple speculative queries can be prefetched.

7. RELATED WORK

Data Interaction with New Computing Devices: With the rise in popularity of multi-touch and gesture devices, there have been several works on interactive data exploration with multi-touch and gesture devices [59, 64, 85, 95, 101, 132]. Compared to mouse and keyboard, new computing devices provide better direct manipulation experience [107] and multi-finger interaction (e.g., zoom in / out), etc. GestureDB [95] and dbTouch [85] are multi-touch query interfaces which translate gestures into database queries. Data3 [59] allows the user to manipulate the 3D cube with the Kinect. PanoramicData [132] is a multi-touch interface focusing on the machine learning and data analysis algorithms. Kinetica [101] is a multi-touch interface allowing the user to explore the visualization with physical affordances. SnaptoQuery [64] allows the user to issue range queries with gesture devices and shows that gesture devices are comparable to the mouse and touch devices. There have been commercial products as well, like Tableau Vizable [18]. However, most of these works focus on usability. There are no existing works for capturing the characteristics of new query workloads, investigating possible behavior-driven op-

timizations with new computing devices, and evaluating the performance of databases with those interactive workloads.

Traditional Benchmarking & Optimizations: Benchmarks are used to measure the performance of systems. [54] surveys popular benchmarks and elaborates the need of domain-specific benchmarks. Traditional benchmarks focus on the transaction processing performance, such as TPC [20] and Wisconsin Benchmark [43]. They typically measure a throughput metric (work/second) and a price metric which is a five-year cost-of-ownership metric. Recently, with the increasing popularity of clouding serving systems (e.g., Hadoop, Spark, etc.), benchmarks are proposed to evaluate their performance. Current cloud benchmarks usually focus on one specific application domain. For example, YCSB [42] focuses on the online service. [2, 40, 41] focus on the real-time analysis. BigDataBench [117] summarizes current status of big data benchmarks and proposes a comprehensive benchmark spans offline analytics, online service, and real-time analytics. Graph and RDF benchmarks are also been proposed, e.g., LinkBench [25], a benchmark based on the Facebook social graph. The Linked Data Benchmark Council (LDBC) [10] was founded a couple of years ago to establish benchmarks for evaluating graph data management systems. [47] is a vision paper that elaborates on use cases for interactive data exploration, followed up by the authors in [46], which provides a benchmark for interactive systems. However, these benchmarks and optimizations do not account for the salient features of interactive query interfaces on non keyboard devices, and further, they do not consider user driven optimizations. As visualizations become a popular channel to communicate results, there are similarly some efforts in evaluating the overlap between visualization and data systems [30]. Finally, with the popularity of mobile devices, benchmarks have been proposed to evaluate the data management in mobile systems [72, 76].

8. CONCLUSION AND FUTURE WORK

Interactive workloads present characteristics that are different from traditional workloads, especially for new devices and interfaces. In this paper, we talk about both existent and emergent characteristics of interactive workloads: difference between devices and interfaces, continuous actions, sensitivity and jitter of gestural devices, and session behavior for adjacent queries. We demonstrate *behavior-driven* optimizations through three case studies which span different interaction devices (mouse, touch, Leap Motion), query interfaces (e.g., map, slider), interaction techinques (e.g., linking & brushing), and queries (e.g., `select`, `aggregation`). Our behavior analyses and performance experiments show that the behavior analyses can help system designers to build more responsive UIs and highly performant backend systems.

There are several directions for future work. We introduced a new metric, latency constraint violation, to measure if a user perceives a delay from the system, and demonstrated it in two of our case studies. However, adapting this metric for other interactive systems requires understanding of the internals of each, and hence a general purpose suite of metric remains an open problem. The metrics discussed in this work concern performance of the interface and backend. The other aspect of evaluating the system involves its information presentation and cognitive effect on the user.

For example, a system that provides *too much* information to the user may be considered performant from the system's standpoint, but could be considered suboptimal from a user's standpoint. In our case studies, this applies to the skip and result based KL divergence optimizations in crossfiltering. We need to conduct user studies to see if users loose information due to queries being skipped. This is especially true in the KL divergence case because if the user is looking for anomalies, the slight difference in result set might be important. Hence, we have to work on methods for measuring loss of information. Orthogonally, if an interface is not designed well, it can lead to the user being disoriented and cognitively overloaded as reported in [102]. There are various standard methods to draw from the area of cognitive science; methods range from measuring the task productivity of a user in the presence of a secondary task in addition to the primary interface query task [110], or measuring the user's physiology using methods such as fMRI [113] and galvanic skin response [96]. However, articulating the direct connection between user's cognitive load and performance aspects of a system involves a large number of confounding factors. Being able to connect the dots by compensating for these confounding factors remains a promising area of future work.

Acknowledgments

The authors would like to thank the conference reviewers for feedback that helped shape this paper. The authors would also like to thank Remco Chang, Aditya Parameswaran, and their students for valuable comments that helped improve early versions of this work. This work is supported by the U.S. National Science Foundation through awards IIS-1422977, IIS-1527779, CAREER IIS-1453582.

9. REFERENCES

[1] Airbnb: Vacation Rentals, Homes, Experiences and Places. `https://www.airbnb.com/`.
[2] AMP Benchmarks. `https://amplab.cs.berkeley.edu/benchmark/`.
[3] Crossfilter Library. `http://square.github.io/crossfilter/`.
[4] Delta. `https://developer.mozilla.org/en-US/docs/Web/Events/mousewheel`.
[5] F 015 Luxury in Motion Concept Car: Interaction with the vehicle through gestures, eye tracking and high-res touch-screens. `https://www.mbusa.com/mercedes/future/model/model-All_New_F015_Luxury`.
[6] Global shipment forecast for touch-screen displays from 2012 to 2016 (in billion units). `https://www.statista.com/statistics/259983/global-shipment-forecast-for-touch-screen-displays/`.
[7] GlobalInterpreterLock. `https://wiki.python.org/moin/GlobalInterpreterLock`.
[8] Google Charts. `https://developers.google.com/chart/`.
[9] IMDb. `http://www.imdb.com/`.
[10] LDBC: The graph and RDF benchmark reference. `http://ldbcouncil.org/benchmarks`.
[11] MemSQL: The Fastest In-Memory Database. `http://www.MemSQL.com/`.

[12] Mutation Events. https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Mutation_events.
[13] OMDb API - The Open Movie Database. http://www.omdbapi.com/.
[14] PostgreSQL: The world's most advanced open source database. https://www.postgresql.org/.
[15] Power BI. https://powerbi.microsoft.com.
[16] ScrollTop. https://developer.mozilla.org/en-US/docs/Web/API/Element/scrollTop.
[17] Tableau. http://www.tableau.com/.
[18] Tableau Vizable. https://vizable.tableau.com/.
[19] TEDCAS uses the Myo armband to give surgeons gesture control. http://blog.thalmic.com/myo-armband-surgery/.
[20] TPC Benchmark. http://www.tpc.org/.
[21] UCI Repository of Machine Learning Databases. https://archive.ics.uci.edu/ml/datasets.html.
[22] Use Multi-Touch gestures on your Mac. https://support.apple.com/en-us/HT204895.
[23] A. Abouzied, J. Hellerstein, and A. Silberschatz. Dataplay: Interactive tweaking and example-driven correction of graphical database queries. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 207–218, New York, NY, USA, 2012. ACM.
[24] W. Albert and T. Tullis. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.
[25] T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan. Linkbench: A database benchmark based on the facebook social graph. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1185–1196, New York, NY, USA, 2013. ACM.
[26] E. Bakke, D. Karger, and R. Miller. A spreadsheet-based user interface for managing plural relationships in structured data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2541–2550, New York, NY, USA, 2011. ACM.
[27] C. M. Barnum. *Usability testing essentials: ready, set... test!* Elsevier, 2010.
[28] R. C. Basole, T. Clear, M. Hu, H. Mehrotra, and J. Stasko. Understanding interfirm relationships in business ecosystems with interactive visualization. *IEEE Transactions on visualization and computer graphics*, 19(12):2526–2535, 2013.
[29] S. Basu Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 13–22, New York, NY, USA, 2008. ACM.
[30] L. Battle, R. Chang, J. Heer, and M. Stonebraker. Position statement: The case for a visualization performance benchmark. In *2017 IEEE Workshop on Data Systems for Interactive Analysis (DSIA)*, pages 1–5, Oct 2017.
[31] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. pages 1363–1375, 2016.
[32] D. A. Bell, L. S. Deluca, D. J. Levinson, and R. Salem. Browser interaction for lazy loading operations, Dec. 15 2014. US Patent App. 14/570,430.
[33] J. Bernard, N. Wilhelm, B. Krüger, T. May, T. Schreck, and J. Kohlhammer. Motionexplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2257–2266, Dec. 2013.
[34] C. Binnig, A. Fekete, and A. Nandi. Hilda '16: Proceedings of the workshop on human-in-the-loop data analytics. New York, NY, USA, 2016. ACM.
[35] A. Biswas, S. Dutta, H.-W. Shen, and J. Woodring. An information-aware framework for exploring multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2683–2692, Dec. 2013.
[36] N. Cao, D. Gotz, J. Sun, and H. Qu. Dicon: Interactive visual analysis of multidimensional clusters. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2581–2590, Dec 2011.
[37] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. In *2008 IEEE Symposium on Visual Analytics Science and Technology*, pages 59–66, Oct 2008.
[38] P. Chau, J. Vreeken, M. van Leeuwen, D. Shahaf, and C. Faloutsos. Proceedings of the acm sigkdd workshop on interactive data exploration and analytics. In *ACM SIGKDD 2016 Full-day Workshop on Interactive Data Exploration and Analytics*. IDEA'16, 2016.
[39] S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histogram construction: How much is enough? In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 436–447, New York, NY, USA, 1998. ACM.
[40] Y. Chen, S. Alspaugh, D. Borthakur, and R. Katz. Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 43–56, New York, NY, USA, 2012. ACM.
[41] 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.
[42] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 143–154, New York, NY, USA, 2010. ACM.
[43] D. J. DeWitt. The wisconsin benchmark: Past, present, and future, 1993.
[44] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 517–528, New York, NY, USA, 2014. ACM.

[45] P. R. Doshi, E. A. Rundensteiner, and M. O. Ward. Prefetching for visual data exploration. In *Eighth International Conference on Database Systems for Advanced Applications, 2003*, pages 195–202, March 2003.

[46] P. Eichmann, C. Binnig, T. Kraska, and E. Zgraggen. Idebench: A benchmark for interactive data exploration. *CoRR*, abs/1804.02593, 2018.

[47] P. Eichmann, E. Zgraggen, Z. Zhao, C. Binnig, and T. Kraska. Towards a benchmark for interactive data exploration. *IEEE Data Eng. Bull.*, 39:50–61, 2016.

[48] J. Faith. Targeted projection pursuit for interactive exploration of high- dimensional data sets. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 286–292, July 2007.

[49] L. Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3):379–383, Aug 2003.

[50] 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.

[51] 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*, 19(12):2149–2158, Dec. 2013.

[52] D. Fisher, I. Popov, S. Drucker, and m. schraefel. Trust me, i'm partially right: Incremental visualization lets analysts explore large datasets faster. ACM Conference on Human Factors in Computing Systems, May 2012.

[53] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. volume 27, pages 261–298, New York, NY, USA, Sept. 2002. ACM.

[54] J. Gray. *Benchmark Handbook: For Database and Transaction Processing Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[55] N. R. Gujarathi and A. A. Shah. Parameterized computed scrolling for navigation of structured data, 2015.

[56] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Approximating multi-dimensional aggregate range queries over real attributes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 463–474, New York, NY, USA, 2000. ACM.

[57] J. Heer, M. Agrawala, and W. Willett. Generalized selection via interactive query relaxation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 959–968, New York, NY, USA, 2008. ACM.

[58] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Acm Sigmod Record*, volume 26, pages 171–182. ACM, 1997.

[59] S. Hirte, A. Seifert, S. Baumann, D. Klan, and K.-U. Sattler. Data3 – a kinect interface for olap using complex event processing. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 1297–1300, Washington, DC, USA, 2012. IEEE Computer Society.

[60] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 277–281, New York, NY, USA, 2015. ACM.

[61] T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, pages 139–148, New York, NY, USA, 2000. ACM.

[62] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 13–24, New York, NY, USA, 2007. ACM.

[63] W. Javed, S. Ghani, and N. Elmqvist. Gravnav: Using a gravity model for multi-scale navigation. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 217–224, New York, NY, USA, 2012. ACM.

[64] L. Jiang and A. Nandi. Snaptoquery: Providing interactive feedback during exploratory query specification. *Proc. VLDB Endow.*, 8(11):1250–1261, July 2015.

[65] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *2014 IEEE 30th International Conference on Data Engineering*, pages 472–483, March 2014.

[66] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3363–3372, New York, NY, USA, 2011. ACM.

[67] J. Kang, J. F. Naughton, and S. D. Viglas. Evaluating window joins over unbounded streams. In *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, pages 341–352, March 2003.

[68] A. Kashyap, V. Hristidis, and M. Petropoulos. Facetor: Cost-driven exploration of faceted query results. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 719–728, New York, NY, USA, 2010. ACM.

[69] A. Kashyap, V. Hristidis, M. Petropoulos, and S. Tavoulari. Effective navigation of query results based on concept hierarchies. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):540–553, April 2011.

[70] M. Kaul, B. Yang, and C. S. Jensen. Building accurate 3d spatial networks to enable next generation intelligent transportation systems. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 137–146, June 2013.

[71] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization*

and Computer Graphics, 8(1):1–8, Jan. 2002.

[72] O. Kennedy, J. Ajay, G. Challen, and L. Ziarek. Pocket data: The need for tpc-mobile. In R. Nambiar and M. Poess, editors, *Performance Evaluation and Benchmarking: Traditional to Big Data to Internet of Things*, pages 8–25, Cham, 2016. Springer International Publishing.

[73] A. Key, B. Howe, D. Perry, and C. Aragon. Vizdeck: Self-organizing dashboards for visual analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 681–684, New York, NY, USA, 2012. ACM.

[74] M. Khan, L. Xu, A. Nandi, and J. M. Hellerstein. Data tweening: Incremental visualization of data transforms. *Proc. VLDB Endow.*, 10(6):661–672, Feb. 2017.

[75] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *Proceedings of the VLDB Endowment*, 8(5):521–532, 2015.

[76] J.-M. Kim and J.-S. Kim. Androbench: Benchmarking the storage performance of android-based mobile devices. In S. Sambath and E. Zhu, editors, *Frontiers in Computer Education*, pages 667–674. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[77] R. Kosara, F. Bendix, and H. Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, July 2006.

[78] A. Kraiss and G. Weikum. Integrated document caching and prefetching in storage hierarchies based on markov-chain predictions. *The VLDB Journal*, 7(3):141–162, Aug 1998.

[79] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[80] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Empirical studies in information visualization: Seven scenarios. *IEEE transactions on visualization and computer graphics*, 18(9):1520–1536, 2012.

[81] H. Lam, D. Russell, D. Tang, and T. Munzner. Session viewer: Visual exploratory analysis of web session logs. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 147–154, Oct 2007.

[82] J. Lazar, J. H. Feng, and H. Hochheiser. *Research Methods in Human-Computer Interaction*. Wiley Publishing, 2010.

[83] D. H. Lee, J. S. Kim, S. D. Kim, K. C. Kim, K. Yoo-Sung, and J. Park. Adaptation of a neighbor selection markov chain for prefetching tiled web gis data. In T. Yakhno, editor, *Advances in Information Systems*, pages 213–222, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[84] Y. Li, H. Yang, and H. V. Jagadish. Nalix: A generic natural language search environment for xml data. *ACM Trans. Database Syst.*, 32(4), Nov. 2007.

[85] E. Liarou and S. Idreos. dbtouch in action database kernels for touch-based data exploration. In *2014 IEEE 30th International Conference on Data Engineering*, pages 1262–1265, March 2014.

[86] 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, Dec. 2013.

[87] B. Liu and H. Jagadish. A spreadsheet algebra for a direct data manipulation query interface. In *2009 IEEE 25th International Conference on Data Engineering*, pages 417–428, March 2009.

[88] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2122–2131, Dec 2014.

[89] Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. 32(3pt4):421–430, 2013.

[90] A. R. Martin and M. O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of the 6th Conference on Visualization '95*, VIS '95, pages 271–, Washington, DC, USA, 1995. IEEE Computer Society.

[91] P. McLachlan, T. Munzner, E. Koutsofios, and S. North. Liverac: Interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1483–1492, New York, NY, USA, 2008. ACM.

[92] D. Moritz, D. Fisher, B. Ding, and C. Wang. Trust, but verify: Optimistic visualizations of approximate queries for exploring big data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2904–2915. ACM, 2017.

[93] F. Moussavi. Hybrid inertial and touch sensing input device, Feb. 18 2010. US Patent App. 12/192,889.

[94] T. Munzner. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, 15(6), 2009.

[95] A. Nandi, L. Jiang, and M. Mandel. Gestural query specification. *Proc. VLDB Endow.*, 7(4):289–300, Dec. 2013.

[96] N. Nourbakhsh, Y. Wang, F. Chen, and R. A. Calvo. Using galvanic skin response for cognitive load measurement in arithmetic and reading tasks. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, pages 420–423, New York, NY, USA, 2012. ACM.

[97] C. Plaisant. The challenge of information visualization evaluation. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 109–116, New York, NY, USA, 2004. ACM.

[98] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 294–305, New York, NY, USA, 1996. ACM.

[99] S. Rahman, M. Aliakbarpour, H. K. Kong, E. Blais, K. Karahalios, A. Parameswaran, and R. Rubinfield. I've seen enough: incrementally improving visualizations to support rapid decision making. *Proceedings of the VLDB Endowment*, 10(11):1262–1273, 2017.

[100] G. M. Rosa and M. L. Elizondo. Use of a gesture user interface as a touchless image navigation system in dental surgery: Case series report. *Imaging science in dentistry*, 44(2):155–160, 2014.

[101] J. M. Rzeszotarski and A. Kittur. Kinetica: Naturalistic multi-touch data visualization. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 897–906, New York, NY, USA, 2014. ACM.

[102] R. G. Saadé and C. A. Otrakji. First impressions last a lifetime: effect of interface type on disorientation and cognitive load. *Computers in human behavior*, 23(1):525–535, 2007.

[103] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Proceedings of the 16th Eurographics Conference on Visualization*, EuroVis '14, pages 351–360, Aire-la-Ville, Switzerland, Switzerland, 2014. Eurographics Association.

[104] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE transactions on visualization and computer graphics*, 18(12):2431–2440, 2012.

[105] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):96–113, July 2005.

[106] 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*, VL '96, pages 336–, Washington, DC, USA, 1996. IEEE Computer Society.

[107] B. Shneiderman, C. Williamson, and C. Ahlberg. Dynamic queries: Database searching by direct manipulation. pages 669–670, 1992.

[108] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. Parameswaran. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *Proceedings of the VLDB Endowment*, 10(4):457–468, 2016.

[109] M. Singh, A. Nandi, and H. V. Jagadish. Skimmer: Rapid scrolling of relational query results. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 181–192, New York, NY, USA, 2012. ACM.

[110] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.

[111] P. Tan. BMW Demonstrates Future iDrive with Touchscreen, Gesture and Tablet Control. *CES 2015*, 2015.

[112] F. Tauheed, T. Heinis, F. Schürmann, H. Markram, and A. Ailamaki. Scout: Prefetching for latent structure following queries. *Proc. VLDB Endow.*, 5(11):1531–1542, July 2012.

[113] L. F. Van Dillen, D. J. Heslenfeld, and S. L. Koole. Tuning down the emotional brain: an fmri study of the effects of cognitive load on the processing of affective images. *Neuroimage*, 45(4):1212–1219, 2009.

[114] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: Automatically generating query visualizations. *Proc. VLDB Endow.*, 7(13):1581–1584, Aug. 2014.

[115] S. D. Viglas, J. F. Naughton, and J. Burger. Maximizing the output rate of multi-way join queries over streaming information sources. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 285–296. VLDB Endowment, 2003.

[116] J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation and histograms via wavelets. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 96–104, New York, NY, USA, 1998. ACM.

[117] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu. Bigdatabench: A big data benchmark suite from internet services. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 488–499, Feb 2014.

[118] C. Weaver. Multidimensional visual analysis using cross-filtered views. In *2008 IEEE Symposium on Visual Analytics Science and Technology*, pages 163–170, Oct 2008.

[119] J. Wei, Z. Shen, N. Sundaresan, and K.-L. Ma. Visual cluster exploration of web clickstream data. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 3–12, Oct 2012.

[120] L. Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006.

[121] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, 2007.

[122] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics*, 22(1):649–658, 2016.

[123] J. Woodring and H.-W. Shen. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):123–137, Jan 2009.

[124] E. Wu, L. Jiang, L. Xu, and A. Nandi. Graphical perception in animated bar charts. volume abs/1604.00080, 2016.

[125] D. Yang, Z. Guo, E. A. Rundensteiner, and M. O. Ward. Clues: A unified framework supporting interactive exploration of density-based clusters in streams. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 815–824, New York, NY, USA, 2011. ACM.

[126] D. Yang, E. A. Rundensteiner, and M. O. Ward. Analysis guided visual exploration of multivariate data. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 83–90, Oct 2007.

[127] J. Yang, A. Patro, S. Huang, N. Mehta, M. O. Ward, and E. A. Rundensteiner. Value and relation display for interactive exploration of high dimensional datasets. In *IEEE Symposium on Information*

Visualization, pages 73–80, 2004.

[128] J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In *Proceedings of the Ninth Annual IEEE Conference on Information Visualization*, INFOVIS'03, pages 105–112, Washington, DC, USA, 2003. IEEE Computer Society.

[129] S. Yeşilmurat and V. İşler. Retrospective adaptive prefetching for interactive web gis applications. *GeoInformatica*, 16(3):435–466, Jul 2012.

[130] J. S. Yi, Y. ah Kang, and J. Stasko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, 13(6):1224–1231, 2007.

[131] X. Yuan, D. Ren, Z. Wang, and C. Guo. Dimension projection matrix/tree: Interactive subspace visual exploration and analysis of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2625–2633, Dec. 2013.

[132] E. Zgraggen, R. Zeleznik, and S. M. Drucker. Panoramicdata: Data analysis through pen & touch. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2112–2121, Dec 2014.

[133] Z. Zhang, K. T. McDonnell, and K. Mueller. A network-based interface for the exploration of high-dimensional data spaces. In *Proceedings of the 2012 IEEE Pacific Visualization Symposium*, PACIFICVIS '12, pages 17–24, Washington, DC, USA, 2012. IEEE Computer Society.