Evaluating Interactive Data Systems: Workloads, Metrics, and Guidelines

Duration: 1.5 hrs

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

Protiva Rahman
The Ohio State University
Columbus, Ohio
rahmanp@cse.ohio-state.edu

Arnab Nandi
The Ohio State University
Columbus, Ohio
arnab@cse.ohio-state.edu

ABSTRACT

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

In this tutorial, we will describe unique characteristics of interactive workloads for a variety of user input devices and query interfaces. We will catalog popular metrics based on an extensive survey of current literature. Through two case studies, we will not only walk through previously defined metrics using real-world user traces but also highlight where these defined metrics are inadequate. Further, we will introduce some new metrics that are required to capture a complete picture of interactivity. In each case study, we also demonstrate how the behavior analyses on users' trace and performance experiments can provide guidelines to help researchers and developers design better interactive data systems.

1 INTRODUCTION

Interactive systems allow the user to clean, visualize and build models for insights almost immediately. This process is often iterative and session based, with the user continuously manipulating subsets of the data. This leads to strict latency requirements since the user's next query is dependent on the results of the previous, and due to the continuous nature, the user can not idly wait. This is a far cry from the old query interfaces where the user types SQL or batches processes that can run overnight, or where the user's queries could be anticipated and optimized for. These factors thus need to be accounted for when benchmarking and optimizing interactive systems.

SIGMOD'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06…$15.00
https://doi.org/10.1145/3183713.3197386

Scope: In this tutorial, we will cover challenges in interactive querying and data exploration, enumerate confounding factors and biases that can create issues with evaluations, survey benchmarks and metrics in the current literature, walk through two case studies that articulate methods to properly benchmark interactivity, and identify open problems. Each of these is described in detail below.

Intended Audience: This tutorial is intended for database systems researchers and industry practitioners, with a focus on researchers interested in human-in-the-loop data analytics, interactive data exploration, query interfaces, visual querying, mobile and next-generation interaction modalities.

2 CHALLENGES IN INTERACTIVE QUERYING AND DATA EXPLORATION

Interactive workloads triggered by different interaction devices and query interfaces present different challenges from traditional workloads. We summarize these challenges here.

Different Types of Query Generation. Different devices have different frame rates per second, which will directly affect the query issuing frequency (number of queries per second). Moreover, the query workload generated by different interfaces is also different. For example, a single map zoom triggers multiple predicate changes (longitude and latitude) in the `where` clause. The slider widget tends to trigger more intensive workloads (number of queries per second) than the text box widget, since typing is slower than sliding. Thus, when creating interactive workloads, a variety of widgets and devices need to be covered.

Strict Latency Constraints. For gesture/touch based devices [56] and widgets that support continuous actions (e.g., slider, linking and brushing) [22], query specification is a continuous gesture. Due to the fluid nature of this interaction, the generated query workload is heavy. For example, consider the case of the slider widget. To issue a query, the user slides continuously to reach the desired position. If the user interface detects events every 20ms, then every second about 50 queries are issued. Since the action is continuous, we need to make sure the whole process is interactive, which requires a strict latency constraint.

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 the interactions more accurate. However, on gestural devices, due to the absence of friction the interaction process is highly variable and sensitive [35]. 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.

Table 1: Metrics for Data Interaction in Current Literature

type	metric	explanation & citations
query interface	usability	Proxied by query specification or task completion time [7, 9, 11, 21, 32–34, 37, 50, 56, 62, 65, 76, 77, 82], number of iterations or navigation cost [12, 38, 39, 48], miss times [35], number and uniqueness of insights [23, 60, 65, 73], accuracy, ability to complete tasks [11, 17, 31, 42, 50, 65, 76, 77]
	learnability	ability to learn user actions with instructions: [9, 11, 56, 60]
	discoverability	ability to discover user actions without instructions: [35, 53, 56]
	user feedback	survey questions: [17, 31, 32, 36–38, 50, 56, 60, 73], case study [15, 26, 27, 44, 54, 58, 63, 71, 74, 78, 79, 81], suggestions [11, 16, 17, 34, 43, 62, 65, 71]
	behavior analysis	sequences of mouse press-drag-release [34], event state sequence [46]
performance	throughput	transactions / requests / tasks per second: TPC-C, TPC-E [4], [18]
	latency	execution time of the query or frame rate per second: [18, 21, 25, 35, 36, 38, 43, 49, 51, 52]
	scalability	performance with increasing data size, number of dimensions, number of machines: [20, 21, 36, 52]
	cache hit rate	[13, 36, 67]

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.

Dependent Queries. In interactive systems, *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.

3 CONFOUNDING FACTORS AND BIASES

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 [8, 10, 24] 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, due to being familiar with the task. In order to combat this, 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.

4 SURVEY OF BENCHMARKS AND METRICS IN CURRENT LITERATURE

Table 1 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 [35, 36]. Even though both, query interface and performance, are critical for a system's evaluation, we find that only few systems [21, 35, 36, 38, 43] evaluate both facets.

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. This makes it difficult to compare two systems, and every new system has to reimplement any system they want to compare against. However, this process becomes easier if certain principles, such as the following, are practiced when evaluating new systems.

(1) When designing high-performance systems, the system designer should take *behavior-driven* optimizations into consideration. Additionally, the systems should be able to handle interactive workloads for different devices and interfaces.
(2) Metrics should maximize the coverage of query types (e.g., `select`, `join`, `aggregation`) and interaction techniques [41, 64, 72, 80] (e.g., filtering, linking & brushing).
(3) User study tasks should simulate *real* use cases on *real* datasets.
(4) Study population should contain at least 10 users [24, 47].
(5) Evaluations should cover a variety of workloads, e.g., different scenarios, data distributions, data sizes, etc.

We demonstrate these principles by walking through two case studies. We show how behavior analyses can dictate metric selection and optimizations. Experiments are performed on multiple devices and interfaces in order to maximize the coverage of interaction techniques and query types. The behaviors studied and metrics used are summarized in Table 2.

Table 2: Metrics Summary

interface	behavior	metrics
inertial scrolling (Section 5.1)	scrolling speed	latency constraint violation
	no. of back scrolls	latency
crossfiltering (Section 5.2)	sliding behavior	query issuing frequency
	querying behavior	latency constraint violation
		latency

5 CASE STUDIES

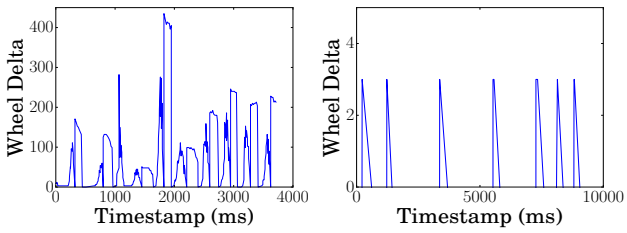
5.1 Inertial Scrolling

Scrolling is a common way to browse information when the result set does not fit on the screen [32, 65]. Inertial or momentum scrolling [55] 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 from the IMDb [2, 3] movie dataset, with 6 attributes, similar to [65]. 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 [6] trackpad. Each user is trained on scrolling on the touchpad before the task.



(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.

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

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 [14, 29] 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 [32, 65].

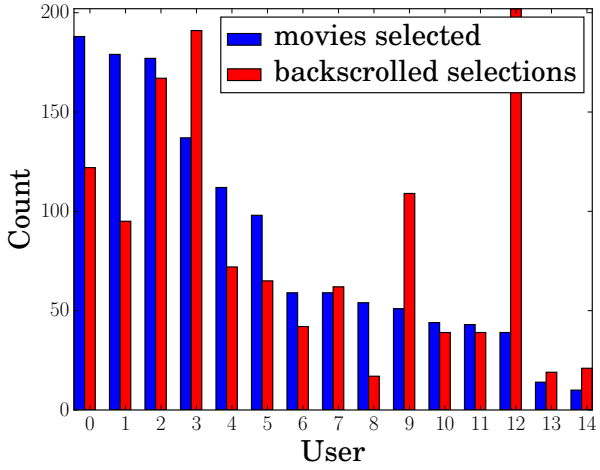


Figure 2: Comparison of number of selected movies and number of movies selected with backscrolls. In some cases, the number of backscrolls is larger than the number of selected movies, indicating that the user had to scroll back and forth multiple times to select a movie. This shows the need for additional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

5.1.2 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 ~ 80 ms. In *timer fetch*, when number of tuples fetched is low, faster users can wait up to ~ 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.

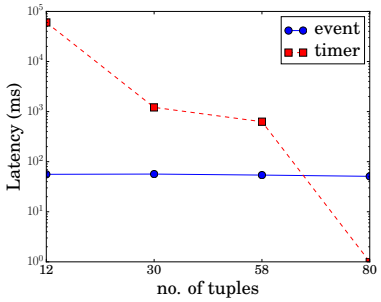


Figure 3: Average latency of 15 users over different number of tuples fetched. *Event fetch* is insensitive to the number of tuples. *Timer fetch* decreases linearly reaching zero latency at median of max scrolling speed.

5.2 Crossfiltering

Crossfiltering (or brushing-and-linking [41]) 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 [1, 35, 49, 52, 70]. Figure 4 shows the UI for our experiments.
Dataset: We used a 3D road network dataset from the UCI ML repository [5, 40], which has 3 attributes 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 [1] was used, which can support extremely fast (<30ms) interactions for up to one million records. Hence, users did not perceive any query latency.

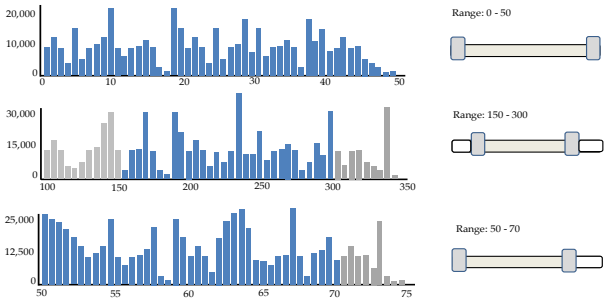
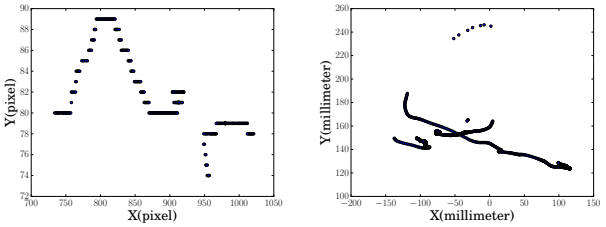


Figure 4: Crossfiltering Interface. Each histogram corresponds to one attribute and range slider. The highlighted region indicates the filtered range for each attribute. Histograms for other attributes are updated synchronously while the user is manipulating one slider.

5.2.1 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. 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.



(a) Mouse **(b) Leap Motion**

Figure 5: Traces of a user specifying a range query on two devices: Leap motion presents more jitter than the mouse.

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 query interface that has high query issuance 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 [30], sampling-based methods [19, 28, 59], wavelets-based methods [69] etc. can be used for fast approximation of histograms i.e. result sets, without running the query. We use the Kullback-Leibler (KL) [45] divergence to measure the difference between two histograms. 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 [75], the authors present initial work on measuring human's perception error of with KL, which can be used to determine the KL threshold.

5.2.2 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

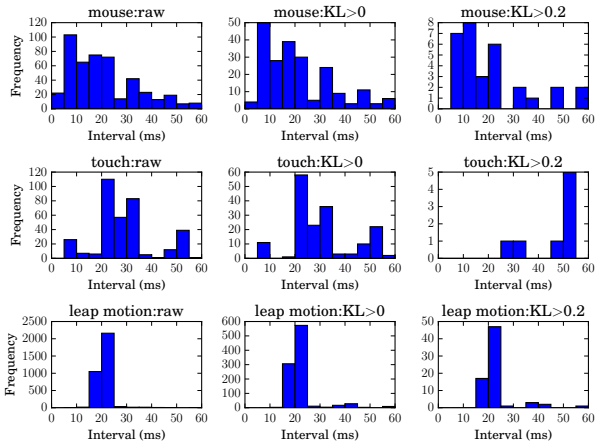


Figure 6: Frequency Histograms of Query Issuing Intervals

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).

6 OPEN PROBLEMS

While the metrics discussed in the previous sections allow us to measure various aspects of interactivity, we explore some additional metrics that would allow us to provide a more complete picture of interactivity for the end-user. In the following paragraphs, we discuss *latency constraint violations* which captures aspects of latency on a more holistic level, and *cognitive load*, which takes the user's abilities into account.

6.1 Latency Constraint Violations

We used latency as a metric for measuring user wait times in both case studies. However, in an interactive system, many queries are not issued in isolation: it can be dependent on the result of another

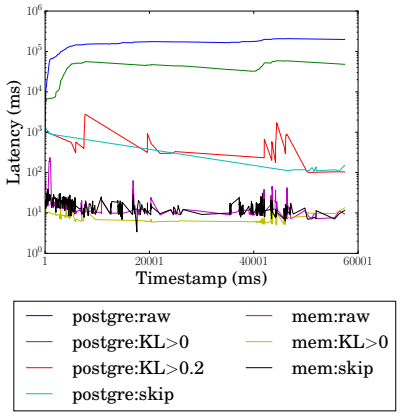


Figure 7: Latency for Mouse under Different Factors. MemSQL can maintain a latency of 10~50ms. After some optimization (KL=0.2 or skip), PostgreSQL can maintain a latency 100~1000ms.

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 [51]). 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 present results of this metric for our two case studies.

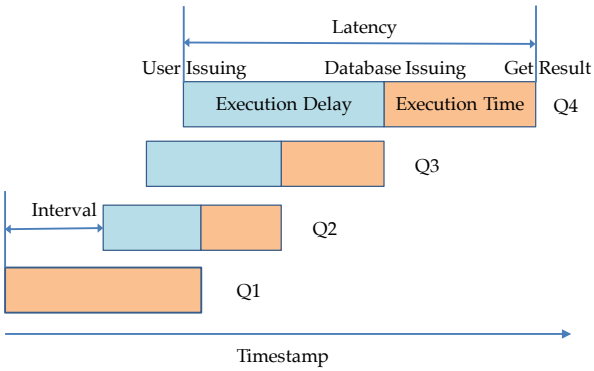


Figure 8: Latency Constraint Violation: Before Q1 finishes its execution, Q2, Q3 and Q4 are already issued by the user in the front-end. When executing Q4, there is already the execution delay caused by previous queries.

6.1.1 Inertial Scroll. 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 3 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.

Table 3: 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

6.1.2 Crossfiltering. 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. Upon analysis, we find that the running times of the violated queries are between 150-500ms for PostgreSQL, and < 25ms on MemSQL.

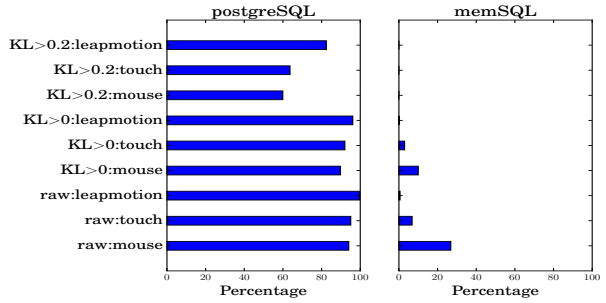


Figure 9: Percentage of Queries that Violate Latency Constraint. KL>0 is enough to improve performance in MemSQL, but KL>.2 is required for observable differences in PostgreSQL.

Thus, latency constraint violation can be used for measuring if a user perceives a delay from the system. Adapting this metric for other interactive systems requires understanding of the internals of each, and hence a general purpose metric remains an open problem.

6.2 Cognitive Load

The metrics discussed so far 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 [61]. 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 [66], or measuring the user's physiology using methods such as fMRI [68] and galvanic skin response [57]. 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.

7 BIOGRAPHIES

Arnab Nandi is an Associate Professor in the Computer Science and Engineering department at The Ohio State University. His research is in the area of database systems, focusing challenges in highly interactive query interfaces and human-in-the-loop data exploration. Arnab is a founding organizer and steering committee member of the Human-in-the-loop Data Analysis and Management (HILDA) workshop. Arnab is a recipient of the US National Science Foundation's CAREER Award, a Google Faculty Research Award, and the Ohio State College of Engineering Lumley Research Award. He is also the 2016 recipient of the IEEE TCDE Early Career Award for his contributions towards user-focused data interaction. Contact information: arnab@cse.osu.edu; (+1 614-292-6377)

Lilong Jiang is a software engineer in the Machine Learning group at Twitter. He received his PhD in Computer Science and Engineering from The Ohio State University in 2017. His dissertation focused on interactive data exploration using gestures, during which he built a variety of interactive data exploration systems such as GestureDB and SnapToQuery. Contact information: jianglil@cse.osu.edu

Protiva Rahman is a PhD student in the Computer Science and Engineering department at The Ohio State University. Her research interests are in the areas of database systems and bioinformatics, with a focus on building novel query interfaces that amplify human expertise. Contact information: rahmanp@cse.ohio-state.edu

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under awards IIS-1422977, IIS-1527779, and CAREER IIS-1453582.

REFERENCES

[1] Crossfilter Library. http://square.github.io/crossfilter/.
[2] IMDb. http://www.imdb.com/.
[3] OMDb API - The Open Movie Database. http://www.omdbapi.com/.
[4] TPC Benchmark. http://www.tpc.org/.
[5] UCI Repository of Machine Learning Databases. https://archive.ics.uci.edu/ml/datasets.html.
[6] Use Multi-Touch gestures on your Mac. https://support.apple.com/en-us/HT204895.
[7] 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*, pages 207–218. ACM, 2012.
[8] W. Albert and T. Tullis. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.
[9] E. Bakke, D. Karger, and R. Miller. A spreadsheet-based user interface for managing plural relationships in structured data. In *SIGCHI*, 2011.
[10] C. M. Barnum. *Usability testing essentials: ready, set... test!* Elsevier, 2010.
[11] R. C. Basole, T. Clear, M. Hu, H. Mehrotra, and J. Stasko. Understanding interfirm relationships in business ecosystems with interactive visualization. *TVCG*, 2013.
[12] S. Basu Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *CIKM*, 2008.
[13] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. 2016.
[14] 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.
[15] 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. *TVCG*, 2013.
[16] A. Biswas, S. Dutta, H.-W. Shen, and J. Woodring. An information-aware framework for exploring multivariate data sets. *TVCG*, 2013.
[17] N. Cao, D. Gotz, J. Sun, and H. Qu. Dicon: Interactive visual analysis of multidimensional clusters. *TVCG*, 2011.
[18] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. In *VAST*, 2008.
[19] S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histogram construction: How much is enough? In *SIGMOD*, 1998.
[20] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *Socc*, 2010.
[21] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *SIGMOD*, 2014.
[22] P. R. Doshi, E. A. Rundensteiner, and M. O. Ward. Prefetching for visual data exploration. In *DASFAA*. IEEE, 2003.
[23] J. Faith. Targeted projection pursuit for interactive exploration of high-dimensional data sets. In *IV*, 2007.
[24] L. Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods*, 2003.
[25] J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *INFOVIS*, 2002.
[26] 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. *TVCG*, 2013.
[27] D. Fisher, I. Popov, S. Drucker, et al. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *SIGCHI*, 2012.
[28] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *VLDB*, 1997.
[29] N. R. Gujarathi and A. A. Shah. Parameterized computed scrolling for navigation of structured data, 2015.
[30] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Approximating multi-dimensional aggregate range queries over real attributes. In *SIGMOD Record*, 2000.
[31] J. Heer, M. Agrawala, and W. Willett. Generalized selection via interactive query relaxation. In *SIGCHI*, 2008.
[32] T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *UIST*, 2000.
[33] H. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *SIGMOD*, 2007.
[34] W. Javed, S. Ghani, and N. Elmqvist. Gravnav: Using a gravity model for multi-scale navigation. In *Advanced Visual Interfaces*, 2012.
[35] L. Jiang and A. Nandi. Snaptoquery: providing interactive feedback during exploratory query specification. *VLDB*, 2015.
[36] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *ICDE*, 2014.
[37] 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*, pages 3363–3372. ACM, 2011.
[38] A. Kashyap, V. Hristidis, and M. Petropoulos. Facetor: cost-driven exploration of faceted query results. In *CIKM*, 2010.
[39] A. Kashyap, V. Hristidis, M. Petropoulos, and S. Tavoulari. Effective navigation of query results based on concept hierarchies. *TKDE*, 2011.
[40] M. Kaul, B. Yang, and C. S. Jensen. Building accurate 3d spatial networks to enable next generation intelligent transportation systems. In *MDM*, 2013.
[41] D. A. Keim. Information visualization and visual data mining. *IEEE transactions on Visualization and Computer Graphics*, 2002.
[42] 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*, pages 681–684. ACM, 2012.
[43] M. Khan, L. Xu, A. Nandi, and J. M. Hellerstein. Data tweening: incremental visualization of data transforms. *Proceedings of the VLDB Endowment*, 10(6):661–672, 2017.
[44] R. Kosara, F. Bendix, and H. Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *TVCG*, 2006.
[45] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.
[46] H. Lam, D. Russell, D. Tang, and T. Munzner. Session viewer: Visual exploratory analysis of web session logs. In *VAST*, 2007.
[47] J. Lazar, J. H. Feng, and H. Hochheiser. *Research methods in human-computer interaction*. John Wiley & Sons, 2010.
[48] Y. Li, H. Yang, and H. Jagadish. Nalix: A generic natural language search environment for xml data. *TODS*, 2007.
[49] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *TVCG*, 2013.
[50] B. Liu and H. Jagadish. A spreadsheet algebra for a direct data manipulation query interface. In *ICDE*, 2009.
[51] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *TVCG*, 2014.
[52] Z. Liu, B. Jiang, and J. Heer. imMens: Real-time Visual Querying of Big Data. *EuroVis*, 2013.
[53] A. R. Martin and M. O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Visualization*, 1995.
[54] P. McLachlan, T. Munzner, E. Koutsofios, and S. North. Liverac: interactive visual exploration of system management time-series data. In *SIGCHI*, 2008.
[55] F. Moussavi. Hybrid inertial and touch sensing input device, Feb. 18 2010. US Patent App. 12/192,889.
[56] A. Nandi, L. Jiang, and M. Mandel. Gestural query specification. *VLDB*, 2013.
[57] 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*, pages 420–423. ACM, 2012.
[58] C. Plaisant. The challenge of information visualization evaluation. In *AVI*, 2004.
[59] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *SIGMOD Record*, 1996.
[60] J. M. Rzeszotarski and A. Kittur. Kinetica: naturalistic multi-touch data visualization. In *CHI*, 2014.
[61] 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.
[62] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Computer Graphics Forum*, 2014.
[63] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information visualization*, 2005.
[64] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Symposium onâĂŹVisual Languages*, 1996.
[65] M. Singh, A. Nandi, and H. Jagadish. Skimmer: rapid scrolling of relational query results. In *SIGMOD*, 2012.
[66] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.
[67] F. Tauheed, T. Heinis, F. Schürmann, H. Markram, and A. Ailamaki. Scout: prefetching for latent structure following queries. *VLDB*, 2012.
[68] 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.
[69] J. S. Vitter, M. Wang, and B. Iyer. Data cube approximation and histograms via wavelets. In *CIKM*, 1998.
[70] C. Weaver. Multidimensional visual analysis using cross-filtered views. In *VAST*, 2008.
[71] J. Wei, Z. Shen, N. Sundaresan, and K.-L. Ma. Visual cluster exploration of web clickstream data. In *VAST*, 2012.
[72] L. Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006.
[73] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *TVCG*, 2007.
[74] J. Woodring and H.-W. Shen. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. *TVCG*, 2009.
[75] E. Wu, L. Jiang, L. Xu, and A. Nandi. Graphical perception in animated bar charts. In *arXiv*.

[76] 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 *CIKM*, 2011.

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

[78] 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 *INFOVIS*, 2004.

[79] 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 *INFOVIS*, 2003.

[80] J. S. Yi, Y. ah Kang, and J. Stasko. Toward a deeper understanding of the role of interaction in information visualization. *TVCG*, 2007.

[81] X. Yuan, D. Ren, Z. Wang, and C. Guo. Dimension projection matrix/tree: Interactive subspace visual exploration and analysis of high dimensional data. *TVCG*, 2013.

[82] Z. Zhang, K. T. McDonnell, and K. Mueller. A network-based interface for the exploration of high-dimensional data spaces. In *PacificVis*, 2012.