Research Statement and Agenda

Eugene Wu
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
eugenewu@mit.edu

"Big Data" is a very real problem, and as every organization embraces data-driven decision making, it will be crucial to develop tools that organize, simplify, and automate this process. However, such wide-spread adoption means that database users cannot be expected to broadly nor deeply understand data management techniques. Instead, they will be domain experts that need simple ways to express and manage complex analyses. This fundamental shift must be addressed at every step in the analysis pipeline — from loading raw data [4], improving core query performance [2, 5, 8], integrating new forms of data [6, 3, 10], helping the user understand anomalies in their results [9], and extending database techniques to support visual interaction [7].

My research approach emphasizes end-to-end systems building that range from low level database query processing such as index and storage layout optimization to designing intuitive query and visual interfaces. I believe that taking a comprehensive view is necessary because bottlenecks can arise at any level and no single technique can sufficiently address them all.

Research Projects

In my thesis, I developed a primitive in data management systems (DBMSs) that can track the records that contribute to a given output result. This is particularly important in exploratory settings where data scientists run a complex array of transformations that reformat and clean input data, rescale dimensions, compute summaries using different algorithms, and combine outputs. When the analyst identifies trends or outliers in the result, it is crucial be able to ask how those results were computed. Tracking such provenance information is expensive, as many common statistical summaries depend on a large number of input data elements and if implemented naively, the overhead of storing and querying the provenance is impractical. Similarly, provenance results can easily innundate the analyst with non-informative inputs (e.g., the sum over an entire table would return every record). Thus, although the concept of provenance has been explored in database systems for several years, general purpose provenance systems remain widely unused in in practice.

To address these limitations, I explored the three key dimensions — overhead, latency, and query result quality — that must be tackled to make provenance systems practical and developed research systems to explore each dimension. Subzero [10] is a DBMS that reduces provenance overhead by tuning the storage behavior based on user-specified runtime and storage constraints. Scorpion [9] is an outlier explanation framework that summarizes the most influential inputs that generated an aggregate outlier. Finally, Smoke [7] is a provenance system for interactive scenarios that can execute provenance queries with low-latency.

SubZero: A Low Overhead Provenance System

Scientific applications such as the Large Synoptic Survey Telescope (LSST) are difficult domains for existing provenance systems because they 1) demand pixel level provenance (e.g., "what pixels generated this star?") for debugging and result validation, 2) are high throughput systems (LSST processes 2GB/sec each night), and 3) can only incur a fixed amount of storage and runtime overhead (LSST budgets $\leq$20% of storage for provenance and must process each image in 15 seconds). Furthermore,

many operators are vectorized, so even generating provenance information can slow operators by orders of magnitude. Finally, custom "black box" operators are common, so the provenance system must provide an efficient APIs to extract the operator's provenance information.

I built SubZero [10], a provenance-enabled workflow system that separates *how* provenance is stored from decisions of *what* provenance to generate. We developed APIs that differentiate operators by the amount of provenance they need to store (constant, linear, or polynomial with the output dataset size). This allows operators to incur the cost of provenance generation and storage in proportion to its complexity. When SubZero executes a provenance query, it can use previously stored provenance or dynamically generate it by re-running previous operators. This lets the optimizer make policy decisions that trade off between provenance query performance and the amount of overhead necessary to achieve such query performance by deciding which operators should generate provenance and what their encoding and indexing strategies should be. In our experiments, SubZero reduced storage overhead and query latency by several orders of magnitude as compared to existing provenance storage models, which is a huge leap towards making provenance systems feasible for high-throughput applications.

Scorpion: Using Provenance to Explain Outliers

When confronted with outlier results, we will naturally ask "why?". Consider a simple analytic query that computes a hospital's total expenses by disease and shows that lung cancer cases disproportionately account for millions of dollars. Is it because those patients tend to need expensive treatments, or is the cause correlated with some other factor of the data? A provenance system can automatically identify all of the lung cancer patients, however an analyst must still manually split the data along different dimensions (e.g., treatment type, age), re-run the query on each subset, and hope one of them points to the cause. If there is more than one outlier, or many dimensions in the dataset, this ad-hoc process becomes untenable. In fact, our Harvard Medical School collaborator spent six months manually analyzing this lung cancer problem. With the Scorpion system, we can identify the two doctors who over-treated their patients and are responsible for a significant amount of the costs within a few minutes of visual interaction and computation.

Scorpion is an example of an interactive system that uses and filters provenance results to answer these "why" questions in the context of aggregation queries. Users simply select outlier and normal results through a novel interface and the system then constructs predicates that most influenced the outliers. Our work formalized this notion of predicate influence in terms of sensitivity analysis, provided a framework to search for influential predicates, and identified aggregation operator properties that help reduce search times by orders of magnitude as compared to a naive exhaustive algorithm.

Smoke: Low Latency Provenance for Interactive Visualizations

Increasingly, people are publishing data as interactive visualizations. Although tools to create static visualizations and animations are prevalent, there are few tools that help create rich interactions — many authors manually implement and optimize interactions such as brushing and linking [1].

Smoke establishes the parallel between common forms of visual interaction and data provenance by modeling visualizations as workflows from raw data to visual elements and visual interactions as provenance queries. This allows visualizations to leverage performance optimizations in provenance systems and scale interactions to larger datasets. I am currently building Smoke, which uses two key insights to execute provenance queries at interactive (<100ms) speeds. First, when designing a publishable visualization, the author can specify the exact set of provenance queries that the system will need to run, which Smoke can optimize its storage representation towards In constrast, existing

[1] A common form of brushing and linking [1] is when a user selects (brushes) a set of points in one view, and the corresponding (linked) objects in other views are also selected

systems are designed for ad-hoc queries and cannot compare the benefits of different optimizations. Second, many interactions do not need all of the source data specified by a provenance query, and only need a sample of the data, their identifiers, or a summary statistic. Smoke uses materialization and approximation techniques to execute these restricted queries must faster.

Research Agenda

I am interested in expanding the ways that data management simplifies data analysis and exploring the intersection between databases, visualization, and interaction. The following are examples of possible projects that I find exciting. The first set of projects extend my thesis work in provenance, while the second are data analytic projects described in the context of an integrated database and visualization system.

Practical Provenance

Scorpion was a first step to show that provenance can explain aggregation query results by properly filtering and prioritizing the provenance information. However, we are still far from using provenance in a general manner to debug and understand our analysis results. The following are some immediate directions to continue this exploration:

Extending Scorpion

Although Scorpion is a general framework, we have only developed optimizations for simple aggregation operators such as MEAN and STDDEV. Extending efficient support to more complex aggregation operators (e.g., linear regression) will require finding additional operator properties that can be leveraged. Additionally, supporting other output types such as data constraint violations (e.g., identifying records that violate a constraint such as "Canada is not a U.S. state") is valuable for data cleaning and intergration.. I plan on pursuing both of these directions.

Understanding Provenance Usage

The Open Provenance Challenge provides a benchmark for evaluating inter-operability between provenance systems, however there are currently no performance-oriented provenance benchmarks. Furthermore, there does not exist a corpus of provenance queries that are used in practice. Without such benchmarks, it is difficult to evaluate different provenance systems and make consistent progress as a field. I plan to release Smoke, which is built into a full-featured visualization system, and gather usage information about how the visualization and provenance systems are used. This information will be developed into a fine-grained provenance and interaction benchmark.

An Integrated Visualization and Data Management System

A key difficulty in scaling interactive visual analytics to large datasets is maintaining interactive (sub-100 millisecond) latencies. Current approaches decouple the visualization and data management systems via a SQL query interface, where each interaction translates to a SQL query and each query can take seconds or minutes to run. The visualization layer compensates for the high latency by either optimizing for a specific visualization (e.g., interactive heat maps), or caching a subset of the data and replicating query processing functionalities in the client. Developing an integrated system has the potential to scale without giving up functionality nor interactivity and enable exciting exploratory features. It also introduces challenges of optimization across the two layers.

Perceputally-accurate Approximation

The output screen resolution is an example of a visual variable that bounds the user perceivable differences (e.g., pixels in this case). An integrated system can reduce query latencies by approximating results while still minimizing user perceived errors by understanding the granularities of the visual

encodings (e.g., color, x-position, radius). For example, color has very low resolution as compared to y-position, so queries that generate heat-maps can afford greater approximation error than those that compute bar charts. Additionally, the system can quickly generate an initial result by computing an approximation that preserves the most salient features in the visualization (e.g., an upward trend in a line chart). I plan to explore how these perceptual properties can inform query processing optimizations such as sampling, approxmation and filtering.

Speculation and Pipelining
Interaction latency can be further reduced by taking advantage of interaction-level semantics. Each visualization affords a limited number of interactions, which constrains the scope of possible queries. In addition, many interactions (e.g., clicking on the border of a selection box to resize it) are triggered by interactions that take up to several seconds to express. This is an ideal environment to speculatively execute queries and pipeline the execution with the time the user takes to express the queries. The key challenges are developing a language to declaratively specify and detect interactions, and mapping them to optimization hints.

Parallel Exploration
An analyst exploring, for example, a large taxi ridership dataset, will often pick a subset such as a single week and location to rapidly iterate and test hypotheses, and finally, expand the analysis to the full dataset. The system can automatically replicate the analysis on different samples (e.g., the previous week, or a different location) to test the robustness of her results or to identify similar trends. Achieving such a goal requires laying out the data so executing the analysis in parallel is efficient, finding query sharing opportunities to re-use computed results, and aggressive pruning techniques to avoid fulling analyzing unpromising datasets.

Improved Recommendations
Recommending relevant data is an important feature in exploratory data analysis – what visualization cues can be used to improve recommendation quality? For instance, semantics of the specific visualization type (e.g., the trend in a line chart) can augment data-level statistical features when suggesting other subsets of the data. Alternatively, the user's exploration history can be used to prune the possible recommendations to those that the user has expressed interest in. How to create a recommendation system that uses features from different *semantic* levels of the system is an exciting direction of research.

. In addition to statistical features used by traditional data mining algorithms, an integrated system can take advantage of the user's exploration history and other visualization level semantics. For instance, map plots may be interested in map tiles with image similarities while a line plot t-tests while a line plot is used to highlight trends, so the system can recommend other distributions or trends, respectively, that validate or differ from the existing results. Alternatively,

References

[1] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[2] P. Cudre-Mauroux, E. Wu, and S. Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *ICDE*, 2010.

[3] A. Marcus, E. Wu, D. R. Karger, S. R. Madden, R. C. Miller, et al. Crowdsourced databases: Query processing with people. In *CIDR*, 2011.

[4] E. Wu. Dbtruck: Humane data import, October 2012.

[5] E. Wu, C. A. Curino, S. R. Madden, et al. No bits left behind. In *CIDR*, 2011.

[6] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *SIGMOD*. ACM, 2006.

[7] E. Wu and S. Madden. Smoke: Visualization interactions using provenance (in preparation).

[8] E. Wu and S. Madden. Partitioning techniques for fine-grained indexing. In *ICDE*, 2011.

[9] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. In *VLDB*, 2013.

[10] E. Wu, S. Madden, and M. Stonebraker. Subzero: a fine-grained lineage system for scientific databases. *ICDE*, 2013.