Research Statement and Agenda

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

The past decade has seen tremendous growth in both the volume of data and the complexity of analyses that organizations want to perform. Increasingly, these analyses are not performed by experienced database administrators but by domain experts that need simple ways to process their data without worrying about scalability and management limitations. To be useful for these users, new systems must simplify analyses and address performance bottlenecks at every step in the analysis pipeline. My work has focused on many of these bottlenecks — robust data ingestion [7], improving core query performance [3, 8, 11, 4, 2], supporting new forms of data and new domains [9, 5, 6, 1], summarizing and debugging query results [12, 13], and scaling up visualizations [10].

More broadly, my goal is to build scalable and useful data management systems for real people. This is reflected in my research approach, which emphasizes building end-to-end systems. I have tackled performance challenges ranging from low-level core database query performance optimization to speeding up high-level data cleaning and result debugging tasks. Such a comprehensive view is necessary in order to identify and address the most significant bottlenecks throughout the entire process.

Research Projects

Although I've investigated many steps of the analysis pipeline during my graduate career, my thesis research specifically explores practical fine-grained provenance techniques for large-scale data analysis applications so that data scientists can easily understand and debug their results. In contrast to systems that track the files and scripts used in a workflow, my work tracks input and output data at the *record* granularity. This is particularly important in exploratory settings where data scientists run a complex sequence of transformations that reformat, clean, rescale, aggregate and combine the input data. When the analyst identifies trends or outlier results — often the most interesting parts of the dataset! — it is crucial to understand where those results came from.

Tracking such provenance information is expensive, as most statistical summaries commonly depend on a large number of input records. Although prior work developed closed-form methods to compute provenance for common SQL operators, it either did not support custom operators, or provided APIs that incurred impractical amounts of storage and query overhead. Moreover, provenance results can easily inundate the analyst with uninformative inputs (e.g., a provenance query on a summation result over the entire table would return every record); previous work has not explored the problem of reducing the provenance to a meaningful subset.

To address these limitations, I developed research systems to explore the three key dimensions that must be tackled to make provenance usage practical — result understandability, overhead, and latency. Scorpion [12] is an outlier explanation framework that summarizes the most influential inputs that generated an aggregate outlier. Subzero [13] is a database that reduces provenance overhead by tuning the storage behavior based on user-specified runtime and storage constraints. Smoke [10] is a provenance system for interactive scenarios that can execute provenance queries with low-latency.

Scorpion: Using Provenance to Explain Outliers

When confronted with outlier results, we naturally ask "why?" Consider the simple analytic query our Harvard Medical School (HMS) collaborator ran that computes a hospital's total expenses by disease. He found that that lung cancer cases disproportionately account for millions of dollars – do these patients require expensive treatments, or is this cost correlated with other factors? While a provenance system can automatically identify the lung cancer as the expensive cohort, an analyst wishing to know why must

still manually partition that data along various combinations of dimensions (e.g., treatment, age), re-run the query on each subset, and hope a combination points to the cause. This ad-hoc process becomes untenable in the presence of multiple outliers, or high dimensionality.

While it took our HMS collaborator six months to analyze the lung cancer problem described above, Scorpion 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 interactive system that uses and filters provenance results to answer "why" questions in the context of SQL aggregation queries. Users simply select outlier and normal results through a novel interface and the system creates filters for potential subgroups of the data that most influenced the outliers. Our work formalized a 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.

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., "which 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). Many operators are vectorized, so even generating provenance information can slow operators by orders of magnitude. Custom "black box" operators are common, so the provenance system must provide efficient APIs to extract the operator's provenance information.

I built SubZero [13], a provenance-enabled workflow system that separates *how* provenance is stored from decisions of *what* provenance to generate. SubZero provides APIs that differentiate operators by the amount of provenance they need to store (constant, linear, or polynomial in the output dataset size). This allows operators to incur the cost of provenance generation and storage in proportion to their 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 provenance query performance and the overhead of achieving such query performance. In our experiments, the combination of efficient encoding schemes and dynamic optimization reduced storage overhead by up to 70× and query costs from a hundred seconds to fractions of a second. This is a huge leap towards making provenance systems feasible in high-throughput applications.

Smoke: Low Latency Provenance for Interactive Visualizations

Data is increasingly published 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 is a system that lets data visualization authors declaratively express visualization interactions as simple provenance queries and automatically optimizes the query execution to maintain low latency.

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 the provenance system and scale interactions to larger datasets. I am currently building Smoke, which uses two key insights to execute provenance queries at interactive ($\leq$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 can be used to optimize its storage representation. In contrast, existing 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 much faster.

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

Research Agenda

I am excited to continue building scalable and usable data management systems and explore the intersection between data management, visualization, and interaction. In particular, I am extending my thesis to explain a larger class of outliers and to develop a provenance benchmark. I will also continue several existing projects for building scalable analysis tools. Finally, I will investigate how visualization and database systems can be co-optimized in an integrated visual analytics system.

Practical Provenance

Scorpion was a first step in showing 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. To push towards practicallity, I will explore the following directions:

"Why" Explanations for More Complex Operators

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 integration. I plan on pursuing both of these directions.

Understanding Provenance Usage

Although the Open Provenance Challenge provides a benchmark to evaluate interoperability between provenance systems, 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. When I release Smoke as a full-featured visualization system, I will gather information about how the visualization and provenance systems are used. This will be the first study on how fine-grained provenance is used in the wild, and the results will be used to develop new provenance interfaces and a fine-grained provenance benchmark.

Core Data Analysis Tools

I intend to continue several projects I started in graduate school that explore scalable data analysis tools.

Robust Data Ingestion

The first step of data analysis is to load the data into the DBMS. However, modern DBMSs ingest data in a single transaction, and expect the input file to be in a consistent, typed and error-free format. A single malformed line can abort the entire loading process, losing all of the progress. DBTruck [7] is a best-effort ingestion tool that automatically parses input files, infers a schema, and bulk-loads the data. DBTruck will detect and log malformed records during the loading process, and gracefully degrade the schema if data-type related errors occur consistently (e.g., alter a float column to a text column). I would like to further develop DBTruck by combining the parsing and type inference steps, integrating the robust loading process into the database system, and extending the automatic data cleaning capabilities once the data has been loaded.

Location Aware Joins

Data is increasingly tagged with location information thanks to the proliferation of data sources such sa sensors, mobile devices, and census surveys. Unfortunately, the location data is encoded at varying degrees of granularity (e.g., a single point per state, or per house), accuracy, and representation (e.g., as a polygon, region, address, or lat/long coordinate). I will continue an earlier project on automatic methods to infer join conditions (e.g., aggregate neighborhood level data by state before joining with state level statistics) and implement the join techniques within the database execution engine. These results can, for example, be used in data exploration and computational journalism to automatically detect correlation relationships within large corpuses of location-based datasets.

A Scalable Visual Analytics System

A key difficulty in scaling interactive visual analytics to large datasets is maintaining interactive (sub-100 millisecond) latencies. This is necessary so analysts can quickly try many different analyses. Current approaches decouple the visualization and data management systems via a SQL query interface — each interaction translates to a SQL query and each query takes seconds or even hours to run. The visualization layer compensates for the high latency by either optimizing for a specific type of visualization (e.g., interactive heat maps), or caching a subset of the data and reimplementing query executors in the client.

I want to develop an integrated exploratory data analysis system where users interact and specify analyses at the visualization level and the query processing system takes advantage of the visualization semantics and human perceptual limitations to execute the analyses with low latency. Such a system has the potential to scale without giving up functionality nor interactivity and to enable valuable exploratory features such as accurate analysis recommendations. The following are several immediate research opportunities.

Perceptually-accurate Approximation

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, human perception of color has very low resolution as compared to position, so queries that generate heat-maps can afford greater approximation error than those that compute bar charts. Similarly, the output resolution is a visual variable that bounds the differences that the user can detect. The system can quickly generate an initial result by computing an approximation that preserves the most perceivable features in the visualization (e.g., an upward trend in a line chart). I plan to explore how these perceptual properties can inform core query processing optimizations such as sampling, approximation and filtering.

Speculation and Pipelining

Interaction latency can be further reduced by taking advantage of interaction-level semantics. Each visualization allows 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 user motions that can take several seconds. 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, mapping them to optimization hints, and taking advantage of the hints.

Parallel Analysis

Analysts commonly test hypotheses on a subset of the data, then apply the analysis to the full dataset. For example, an analyst exploring a large taxi ridership dataset may pick the rides during a single week and location to rapidly iterate upon, and later expand the analysis to the full dataset or over other weeks and locations. The system can automatically replicate the analysis on different samples (e.g., the previous week, or a different location) to test result robustness or to identify similar trends. This requires optimizations deep in the data management layer to layout the data so the analysis can be efficiently parallelized, find query sharing opportunities to reuse computed results, and develop pruning techniques to avoid fully analyzing unpromising datasets.

Contextual Recommendations

Recommending relevant data, visualizations and analyses is an important tool in exploratory data analysis. Prior recommendation algorithms are typically implemented at the database layer and only have access to the data and the historical SQL queries. Higher level cues, such as the analyst's exploration history and visualization-specific features (e.g., distributional differences in a boxplot vs trends in a line plot) can substantially improve the recommendation quality. I would like to explore how features across multiple semantic levels — raw data, query result, visualization, image — can be combined and leveraged. The techniques for tackling *Parallel Analysis* will be equally applicable for quickly generating recommendations.

References

[1] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Uncovering the relational web. In *WebDB*, 2008.

[2] P. Cudré-Mauroux, E. Wu, and S. Madden. The case for rodentstore: An adaptive, declarative storage system. In *CIDR*, 2009.

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

[4] C. Curino, E. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich. Relational Cloud: A Database Service for the Cloud. In *CIDR*, 2011.

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

[6] A. Marcus, E. Wu, D. R. Karger, S. R. Madden, and R. C. Miller. Human-powered sorts and joins. In *VLDB*, 2011.

[7] E. Wu. DBTruck: Humane data import, October 2012. `http://istc-bigdata.org/index.php/dbtruck-humane-data-import/`.

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

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

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

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

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

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