The Case for Interactive Data Exploration Accelerators (IDEAs)

Andrew Crotty Alex Galakatos Emanuel Zgraggen Carsten Binnig Tim Kraska
Department of Computer Science, Brown University
{firstname_lastname}@brown.edu

ABSTRACT

Enabling interactive visualization over new datasets at "human speed" is key to democratizing data science and maximizing human productivity. In this work, we first argue why existing analytics infrastructures do not support interactive data exploration and then outline the challenges and opportunities of building a system specifically designed for interactive data exploration. Finally, we present an Interactive Data Exploration Accelerator (IDEA), a new type of system for interactive data exploration that is specifically designed to integrate with existing data management landscapes and allow users to explore their data instantly without expensive data preparation costs.

1. INTRODUCTION

Truly interactive visualization applications allow users to make data-driven decisions at "human speed," but traditional DBMSs are ill-suited to serve this class of applications. Historically, DBMSs assume (1) long data loading times (e.g., for index construction), (2) text-based input and output (e.g., a SQL terminal), (3) relatively simple OLAP queries, and (4) a one-shot (i.e., stateless) batch querying paradigm, therefore making them an exceptionally bad fit for interactive data exploration. At the same time, the expectation that a new system supporting both data warehousing and interactive data exploration will replace existing data management stacks is, simply, unrealistic. Instead, a system designed specifically for interactive data exploration must integrate and work seamlessly with existing data infrastructures (e.g., data warehouses, distributed file systems, analytics platforms).

In this paper, we outline our vision and present initial results for a new system designed to enable interactive data exploration. We suggest dropping the aforementioned assumptions of traditional DBMSs and propose a new breed of systems that support (1) immediate exploration of new datasets without the need for expensive data preparation, (2) visual input and output, (3) complex analytics tasks like machine learning (ML), and (4) "conversational" user interactions with early results that progressively refine over time. Furthermore, rather than aiming to replace or extend existing systems, our goal is to build the first *Interactive Data Exploration Accelerator* (IDEA) that connects to existing data management infrastructures in order to speed up query processing for visual data exploration tools like Vizdom [8].

Figure 1 shows an example workflow created in Vizdom where a user analyzes data gathered from the 1994 US census dataset [13]. In this example, the goal is to determine which features (shown as boxes on the left) affect whether an individual earns a salary of more than $50k annually. To answer this question, the user first drags out the `sex` attribute to the canvas (Step A) to view the distribution of males and females. The user then drags out the `salary` attribute, links these two visualizations, and selects the female bar to view the filtered salary distribution for females only (Step B). A duplicate of the `salary` visualization connected with a negated link (dotted line) allows a comparison of the relative salaries of males and females (Step C). After some analysis, the user decides to check whether an individual's `education` level coupled with `sex` has an impact on `salary`. Finally, linking `education` to each of the `salary` visualizations and selecting only individuals with a PhD (a rare subpopulation) creates a complex workflow comparing the respective salaries of highly educated males and females (Step D). From this analysis, the data seem to suggest that highly educated females earn less money annually than their male counterparts. To further explore this finding, the user might continue the analysis by testing the impact of additional attributes, applying statistical techniques (e.g., a t-test) to validate the finding, or performing various ML tasks (e.g., classification, clustering) to test other hypotheses. A more complete video demonstration of Vizdom is available at [6].

Throughout this data exploration process, the backend system must consistently provide response times low enough to guarantee fluid user interactions in the frontend. In fact, a recent study [14] shows that even small delays (more than 500ms) significantly decrease a user's activity level, dataset coverage, and insight discovery rate. No existing techniques, though, can guarantee interactive latencies while also addressing all of our previously stated goals. For example, existing data warehouses require a full copy of the data and suffer from long loading times, which contradicts our goal of being able to start exploring a new dataset immediately without expensive data preparation (e.g., indexing, compression). Furthermore, many existing data warehouse indexing techniques suffer from the "curse of dimensionality" and do not scale well beyond a handful of attributes [4]. Restricting the number of attribute combinations is also not an option, since the core idea of data exploration is to look for new, unexplored relationships in the data. Finally, dynamic data reorganization techniques (e.g., cracking [12]) do not solve the high-dimensionality problem and require sorting the data based on user access patterns, thereby violating the key randomness assumption of many online algorithms.

HILDA'16, June 26 2016, San Francisco, CA, USA

ISBN 978-1-4503-4207-0/16/06…$15.00

DOI: http://dx.doi.org/10.1145/2939502.2939513

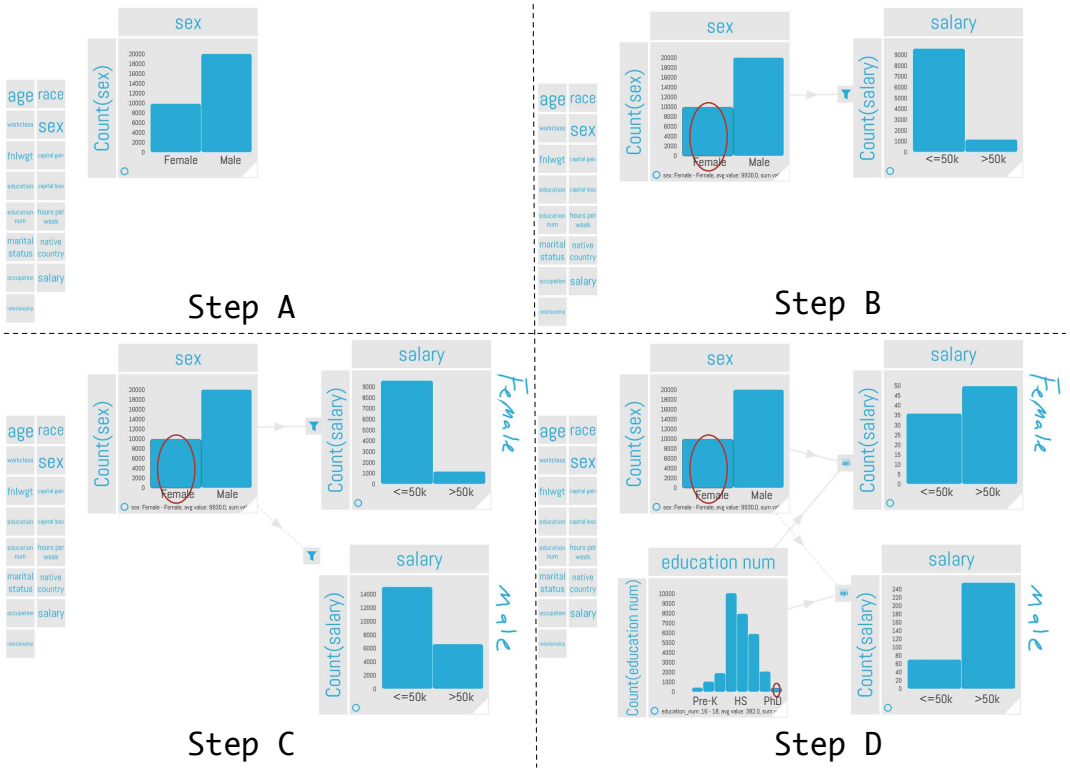

Figure 1: Example Workflow to analyze Salary Distributions

In order to return early results for long-running queries over large datasets, online aggregation techniques [11, 17] provide approximate answers with an estimated error, and several newer analytics frameworks (e.g., Spark [19], Flink [2]) now support online aggregation using a streaming execution model. We therefore believe that online aggregation is a good starting point for Vizdom's backend, since these techniques will allow the system to quickly provide initial results that are refined over time as more data is scanned. However, while online aggregation techniques work well for approximating results to queries on common subpopulations in the data, they start to break down when applied to increasingly rare subpopulations (e.g., when the user in the example selected individuals with a PhD), since scanning the base data in a random order might not provide enough of these instances to provide an accurate estimate. This problem is quite common in many interactive data exploration use cases, where rare events often contain the most interesting insights (e.g., the habits of the few highly valued customers). Although disproportionate stratified sampling can help in these cases by overrepresenting rare data items, these samples typically need to be fully constructed before data exploration even begins [1, 5], contradicting our goal of enabling immediate exploration of new datasets. More importantly, though, most of these systems make the strong assumption that the entire workload is known a priori in order to create appropriate samples, whereas our goal is to allow users to explore data in new and potentially unanticipated directions.

In this paper, we make the case for Interactive Data Exploration Accelerators (IDEAs), which allow users to connect to existing data sources and immediately begin the process of visual data exploration. In summary, we make the following contributions:

- We outline the challenges and opportunities associated with building an IDEA (Section 2).
- We describe the design and unique contributions of the A-WARE system, the first IDEA (Section 3).
- We present our initial results, which show that our prototype can provide interactive response times for different data exploration workflows compared to more traditional approaches (Section 4).

2. CHALLENGES AND OPPORTUNITIES

Designing an accelerator for interactive data exploration with a human-in-the-loop frontend requires solving a set of very unique research challenges while also opening the door to several interesting opportunities. In this section, we first outline some of the requirements and challenges for an IDEA, followed by an overview of some of the unique opportunities to address them.

2.1 Challenges

Interactive data exploration has a very unique set of requirements (e.g., response time guarantees), many of which are pushing the boundaries of what is feasible today.

Interactive Latencies: By far, the most important challenge in supporting interactive data exploration is to display a result within the latency requirement. As [14] showed, even small delays of more than 500ms can significantly impact the data exploration process and the number of insights a user makes. Therefore, IDEAs need to maintain certain response time guarantees in order to provide a fluid user experience.

Rare Data Items: Data exploration often involves examining the tails of a distribution to view the relatively rare data items. For example, real world datasets are rarely perfectly clean and often contain errors (which are typically rare) that can still have a profound effect on the overall results. Similarly, valid outliers and the tails of the distribution are often of particular interest to users when exploring data (e.g., the few billionaires in a dataset, the super users, the top-k customers, the day with the highest traffic). Unfortunately,

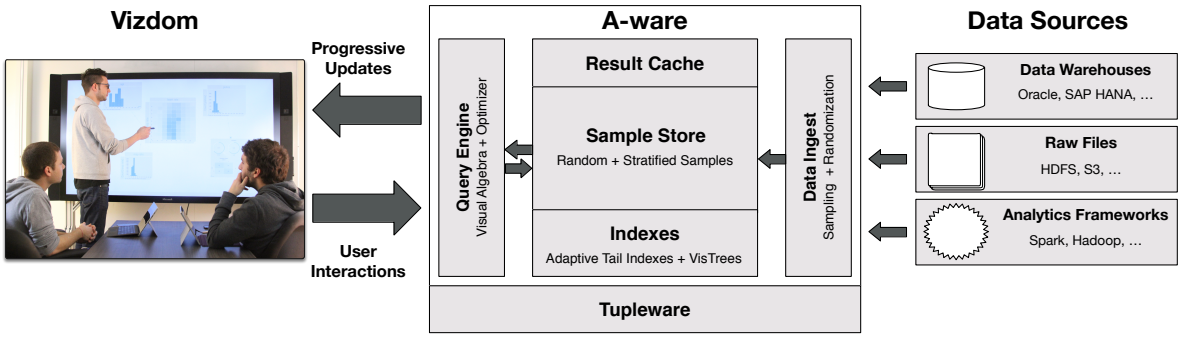

Figure 2: BIDES Architecture Overview

for rare events and the tail of the distribution, sampling techniques do not work well since they often miss rare items or require a priori knowledge of the workload, a challenge when designing an IDEA.

Connect and Explore: Ideally, the user should be able to connect to a dataset and immediately start exploring it. However, this requirement implies that there is no time for data preparation and the system has to build all index structures on the fly. Another implication of the *connect and explore* paradigm is that the system has to stream over larger datasets (from the sources) and may not be able to hold the entire dataset in memory (or even on disk). As outlined in the introduction, online aggregation methods are a good fit to overcome this challenge, since they provide an immediate estimate (with error bars) over the incoming stream. However, online aggregation techniques assume that the data is random, which might be false since some data sources (e.g., data warehouses) often sort the data on some attribute. This can result in a biased estimate of the result and invalid error bars. Similarly, no good estimates are possible if the source returns the data in some chronological order and if there is some (unknown) correlation between time and the value of interest (e.g., the sales are increasing over time).

Interactive ML: ML algorithms can often take a lot of time to build a single model. There has been a lot of work in online machine learning algorithms that can simultaneously update a model using several data items. Yet, the focus of these techniques is usually different. Rather than aiming to provide a quick answer, the goal of online learning is to determine which data item provides the biggest benefit (e.g., to prioritize which data labels to generate next). Furthermore, these techniques are often not the first choice for data scientists during data exploration; the common wisdom is to start with simple techniques first and then add complexity later. Consequently, there is a need for a new set of standard techniques (e.g., basic Lasso Regression, Random Forests, etc.) with interactive latency guarantees.

Consistency: Since fast response times are often only possible by approximating results and parallel computation, inconsistencies in the view can occur. For example, in Figure 1(B), the combined salary bars on the right visualization may not add to the total number of females shown in the left visualization, leading to an inconsistent state. While we initially assumed that inconsistencies pose an important challenge, it turns out that this problem only appears in a few corner cases as explained later in more detail.

Quantifying Risk: A interactive data exploration system with a visual interface allows users to explore hundreds of hypotheses in a very short amount of time. Yet, with every hypothesis test (either in the form of an explicit statistical test or through a more informal visualization), the chance of finding something by chance increases. Additionally, the visual interface can make it easier to overlook other challenges, (e.g., "imbalance of labels" for a classifier) which can lead to incorrect conclusions. Therefore, quantifying the risk is extremely important for an interactive data exploration system.

2.2 Opportunities

Although there are several challenges to address, there are many unique opportunities, since data exploration involves close interactions between analysts and the system. Many of these challenges have not yet been explored within the data management community.

Human Perception: One of the most interesting opportunities stems from the fact that all results are visualized. Therefore, often precise answers are not needed and approximations suffice. Furthermore, the human eye has limitations and humans are particularly bad at understanding the impact of error bars [9]. The system can exploit both of these properties to provide faster response times (i.e., only compute what is perceived by the user).

Think Time: Although the user expects subsecond response times from the system, the system's expectation from the user is different; there might be several seconds (or sometimes even minutes) between user interactions. During this time, the system not only has the chance to improve the current answers on the screen, but also prepare for any future operations. For instance, in our running example, the user might have already dragged out the `sex` and `salary` attribute, but not yet linked them together. Given that both attributes are on the screen, the system might begin creating an index for both attributes. Should the user decide to link the two visualizations and use one as a filter, the index is already created to support this operation.

Interaction Times: Similar to think time, the system can also leverage the user interaction time to provide faster and more accurate answers. For example, it takes several hundred valuable milliseconds to drag an attribute on the interactive whiteboard or to link two visualizations together. Similarly, to adjust the hyperparameters of a ML algorithm, the user first click on the parameter button and then moves a slider. In contrast to the previous think time, user interactions are much shorter but usually provide more information to the system about the intent of the user.

Query Sessions and Reuse: In contrast to one-shot DBMS sessions, data exploration is an iterative, session-driven process where a user repeatedly modifies a workflow after examining the results until finally arriving at some desired insight. This session-driven discovery paradigm provides a lot of potential to reuse results between each interaction and modification.

Data Source Capabilities: Traditional analytics systems like Spark and streaming systems like Streambase assume that they connect to a "dumb" data source. However, many data sources are far from "dumb". For instance, commonly the data source is a data warehouse with existing indexes, materialized views, and many other advanced capabilities. While these capabilities do not

directly fulfill the needs for interactive data exploration, they can still be used to reducing load and network traffic between the data warehouse and the accelerator. Furthermore, there has been work on leveraging indexes [16] to retrieve random samples from a DBMS. These techniques, together with the possibility to push down UDFs to randomize data, provide a feasible solution to the previously mentioned bias problem.

Modern Hardware: Finally, there are several modern hardware trends that can significantly improve the amount of work that can be done in less than 500ms. While there has been already a lot of work in leveraging GPUs for data exploration [15], most of the existing solutions focus on single machine setups and ignore the potential of small high-performance clusters. Small high performance clusters can help to significantly increase the amount of available main memory (1-2 TB of main memory is not uncommon with 8 machine cluster), which is crucial for interactive speeds, while avoiding the problems of fault-tolerance and stragglers that come with large cloud deployments. At the same time, fast network interconnects with RDMA capabilities are not only more affordable for smaller clusters, but also offer unique opportunities to decrease latencies. For example, transferring $2KB$ with TCP/IP over 1GB Ethernet requires $100\mu s$ vs only $2\mu s$ with RDMA and InfiniBand FDR 4x. However, taking full advantage of the network requires carefully redesigning the storage layer of the system in order to enable remote direct memory access [3].

3. THE A-WARE SYSTEM

The A-WARE system is the first IDEA built specifically to enable users to visually explore large datasets through "conversational" interactions. Our prototype addresses many of the previously mentioned challenges (Section 2), applying novel progressive sampling, indexing, and query optimization techniques in order to provide interactive response times. In this section, we first provide an overview of our proposed architecture, followed by highlights of research insights and contributions.

3.1 Architecture

A-WARE is part of Brown's Interactive Data Exploration Stack (BIDES), which is shown in Figure 2. The Vizdom frontend provides a visual data exploration environment specifically designed for pen and touch interfaces, such as the recently announced Microsoft Surface Hub. Figure 2 includes an actual picture of the Microsoft Surface Hub in our lab running Vizdom to explore a medical dataset. A demo video of Vizdom can be found here [6]. Currently, Vizdom connects to A-WARE using a standard REST interface, which in turn connects to the data sources using the appropriate protocols (e.g., ODBC). These data sources can include anything from legacy data warehouses to raw files to advanced analytics platforms (e.g., Spark [20], Hadoop [18]).

As shown in the figure, Vizdom connects to A-WARE, which acts as an intelligent cache and streaming engine that uses Tupleware [7] as a runtime for more complex analytics tasks. Tupleware is a general purpose distributed analytics framework specifically designed for small high-performance clusters, thereby allowing A-WARE to take full advantage of modern hardware (see [7] for more details).

A-WARE roughly divides the memory into three parts: the *Result Cache*, the *Sample Store*, and space for *Indexes*. When triggered by an initial user interaction, A-WARE begins ingesting data from the various data sources, speculatively performing operations and caching the results in the *Result Cache* to support possible future interactions. At the same time, A-WARE also caches all incoming data in the *Sample Store* using a compressed row format. When the available memory for the *Sample Store* is depleted, A-WARE starts to update the cache using a reservoir sampling strategy to eventually create a representative sample over the whole dataset. Furthermore, A-WARE might decide to split up the reservoir sample into several stratified subsamples to overrepresent the tails of the distribution, or to create specialized indexes for potential future queries. All these decisions are constantly optimized based on both past and current user interactions. For example, if the user drags a new attribute onto the canvas, the system might allocate more resources to the dragged attribute and preparation for potential follow-up queries. At the same time, A-WARE constantly streams increasingly precise results to the frontend as the computation progresses over the data, along with indications about both the completeness and current error estimates.

3.2 Research Findings and Contributions

In this section, we highlight a few selected research findings and contributions of A-WARE.

Neither a DBMS nor a Streaming Engine: An IDEA is neither a DBMS nor a streaming engine, instead has an entirely unique semantics. Unlike DBMSs, queries are not one-shot operations that return batch results; rather, workflows are constructed incrementally, requiring fast response times and progressive results that refine over time. At the same time, streaming engines traditionally deploy predefined queries over infinite data streams, whereas an IDEA is meant to enable free-form exploration of data sampled from a deterministic system (e.g., a finite data source). Fundamentally, A-WARE acts as an intelligent, in-memory caching layer that sits in front of the much slower data sources, managing both progressive results and the samples used to compute them. Oftentimes, A-WARE has the opportunity to offload pre-filtering and pre-aggregation operations to an underlying data source (e.g., perform a predicate pushdown to a DBMS), or even transform the base data by executing a custom UDF in an analytics framework.

Finally, in contrast to traditional DBMSs and streaming engines, users compose queries incrementally, therefore resulting in simultaneous visualizations of many component results with varying degrees of error. Maintaining different component partial results rather than single, exact answers imposes a completely new set of challenges for both expressing and optimizing these types of queries. Currently, our A-WARE prototype uses a preliminary interaction algebra to define a user's visual queries.

Visual Indexes: Similar to the algebra and optimizer, we also found that traditional indexes are not optimal for interactive data exploration tasks. Most importantly, existing techniques either sort the data (e.g., database cracking) or do not naturally support summary visualizations. As previously mentioned, sorting can destroy data randomness and, consequently, the ability to provide good estimates. Similarly, indexes generally index every tuple without considering any properties of the frontend (e.g., human perception limitations, visualization characteristics). This approach often results in very large indexes, especially with increasingly large samples or highly dimensional data.

For example, some visualizations (e.g., histograms) require the system to scan all leaf pages in a traditional B-tree, since this index is designed for single range requests rather than providing visual data summaries. We therefore developed VisTrees [10], a new dynamic index structure that can efficiently provide approximate results specifically to answer visualization requests. The core idea is that the nodes within the index are "visually-balanced" to better serve visual user interactions and then compressed based on perception limitations.

Sample Management: As previously mentioned, A-WARE caches as much data as possible from the underlying data sources in order

Interaction	V1	V2	L1	S1	S2	R1	L2	S3	V3	L3	S4	S5	R2	L4	S6
MonetDB	0.476	2.100	0.000	2.000	3.200	0.480	0.000	0.436	2.100	0.549	2.000	0.531	0.610	2.800	1.700
Online Agg.	0.139	0.410	0.000	1.036	0.893	0.142	0.000	4.081	0.458	6.438	1.469	6.382	0.138	4.731	6.932
IDEA	0.140	0.419	0.000	0.492	0.002	0.002	0.000	0.002	0.461	0.597	1.473	0.002	0.002	0.631	0.002

Figure 3: Latencies (in seconds) for Workflow 1

Interaction	V4	L5	S7	R3	L6	V5	L7	L8	N1
MonetDB	0.91	1.63	4.30	2.10	4.20	0.46	2.00	4.10	1.90
Online Agg.	0.14	6.82	7.31	5.91	7.29	0.14	5.86	7.28	6.79
IDEA	0.14	0.66	0.86	0.00	0.00	0.00	0.00	0.00	0.00

Figure 4: Latencies (in seconds) for Workflow 2

to provide faster approximate results, since most data sources are significantly slower. For example, the memory bandwidth of modern hardware ranges from $40 - 50$GB/s per socket [3], whereas we recently measured that PostgreSQL and a commercial DBMS can only export $40 - 120$MB/s, even with a warm cache holding all data in memory. Although DBMS export rates may improve in the future, A-WARE's cache will still remain crucial for providing approximate answers to visual queries and supporting more complex analytics tasks (e.g., ML algorithms).

If the cached data exceeds the available memory, A-WARE needs to carefully evict stored tuples while retaining the most important data items in memory. For example, caching strategies like LRU do not necessarily maintain a representative sample. Therefore, A-WARE uses reservoir sampling instead to evict tuples while preserving randomness. Furthermore, A-WARE also needs to maintain a set of disproportionate stratified samples that overrepresent uncommon data items in order to support operations over rare subpopulations.

The necessity to maintain different types of potentially overlapping samples poses many interesting research challenges. For example, deciding when and what to overrepresent is a very interesting problem, though A-WARE currently employs a greedy algorithm that attempts to create a stratified sample for each operation. Furthermore, A-WARE can even use stratified samples to overcome the label imbalance problem encountered by many ML algorithms (e.g., overrepresenting the rare subpopulation of individuals with a PhD when training a classifier to predict education level). Yet, making these types of decisions can have profound effects on the overall performance, error estimates, and a user's understanding of the model, all of which we are still actively investigating.

Probability Formulation: While developing A-WARE, we observed that many visualizations rely on the observed frequencies in the underlying data, or estimates of the probability of observing certain data items. For example, a bar chart over a nominal attribute is simply a visualization of the relative frequencies of the possible attribute values (i.e., a probability mass function), and a histogram of a continuous attribute visually approximates the attribute's distribution (i.e., a probability density function). Although seemingly trivial, this observation prompted us to reconsider online aggregation as a series of probability expressions. This novel probability formulation actually permits a wide range of interesting optimizations including taking advantage of the Bayes' theorem to maximize the reuse of results. Similarly, our probability formulation can even be leveraged when running some statistical tests or ML algorithms.

Our current implementation of A-WARE therefore manages a cache of results that stores previously computed frequencies and error estimates for reuse in future queries. We plan to expand upon this idea, including the new optimizations enabled by the new formulation, in future work.

Inconsistencies: Interactive response times often require computing approximate answers in parallel, which can lead to inconsistencies in concurrent views (e.g., the combined `salary` bars shown in Figure 1(B) may not sum to the total number of females). Similarly, an outlier that appears in one result visualization may not yet be reflected in another, causing the user to draw a potentially incorrect conclusion.

Although initially assuming that inconsistencies would pose an important challenge for A-WARE, we found that this problem only arises in a few corner cases, and we did not observe any consistency issues during our user study. In particular, A-WARE's result reuse and sampling techniques work together to mitigate many potential consistency problems, and any noticeable differences tend to disappear before the user can even recognize them.

4. INITIAL RESULTS

In this section, we compare A-WARE to other alternatives, including a column-store DBMS (MonetDB 5) and a traditional implementation of online aggregation. We ran all benchmarks on a single dual socket machine with Intel E7-8830 processors (8 cores, 24MB cache), 512GB RAM, and 2TB RAID 10 HDDs. Overall, our benchmarks show that our techniques can almost always deliver results within the human interactivity threshold.

4.1 Data and Workflows

Our experiments use a modified version of the Adult dataset [13], a subset of the 1994 US census. This dataset includes 15 attributes (nine nominal, six continuous), such as sex, education, and a binary label indicating whether an individual earns more than $50k annually. In order to test scalability, we scaled the data size while maintaining the underlying distribution of values. Here, we only show results for the 10GB dataset. For the evaluation, we simulated two workflows, both of which were derived from interaction logs collected during an ongoing user study with 35 participants. In our study, students from the *Intro to Data Science* class at Brown University were able to start using the tool to explore several datasets after only a short introduction to Vizdom.

In the following, we describe the two workflows in more detail and highlight every user interaction in bold text, with **V#** referring to the creation of a new visualization, **L#** linking two visualizations, **S#** selecting an attribute as a filter condition on one visualization, and **R#** removing a link or visualization.

Workflow 1: The goal of the first workflow is to determine how attributes like `sex` and `education` influence an individual's `occupation`. The first step in this workflow is to drag out the `sex` attribute (**V1**) followed by `education` (**V2**). Next, V1 and V2 are linked (**L1**), and the female bar in V1 is selected (**S1**). Then, the selection is toggled to include only males (**S2**) before removing the link (**R1**), which triggers a rerendering of V2. To examine the distribution of males and females for various education levels, V2 and V1 are linked in the opposite direction (**L2**), with only the PhDs selected first (**S3**). Dragging `occupation` to the canvas (**V3**) and linking with education (**L3**) then shows the breakdown of occupations for PhDs. Switching the selection from PhD to HS-grad (**S4**) and then back to PhD (**S5**) enables a quick comparison

of the impact of education on both sex and occupation. Finally, removing L2 and creating a new link from V1 to V3 (**L4**) shows the distribution of occupations for only males with PhDs, which is then compared to the occupations of females with PhDs by switching the selection in V1 (**S6**).

Workflow 2: The second workflow is a continuation of the first, where the goal is to understand how the explored features affect the `salary` attribute (similar to the final workflow shown in step D of Figure 1). After analyzing the impact of `sex` and `education` on `occupation`, the `salary` attribute is dragged onto the canvas (**V4**) and linked with V3, filtering only the salaries for the subset selected in the previous workflow (**L5**). Selecting only the Prof-specialty occupation (**S7**) further filters the displayed salaries. In order to compare males and females, L4 is removed (**R3**), and V1 is re-linked to V4 directly (**L6**). V4 is duplicated by dragging out the `salary` attribute (**V5**), followed by linking both V3 (**L7**) and V1 (**L8**) to the newly created V5. Finally, negating L8 (**N1**) produces a visualization that compares the salaries of male Prof-specialty workers with doctorates with their female counterparts.

4.2 Discussion

Figures 3 and 4 show the latencies for each step in the previously described workflows. Our main design goal is to provide usable (approximate) results within the human interactivity threshold of 500ms. Therefore, each cell in the tables are color-coded to indicate whether the query latency is below the threshold, ranging from green (significantly below the threshold) to red (above threshold). Orange/Yellow values are just around 500ms.

Since MonetDB does not support approximate query answers, the response times in Figures 3 and 4 show how long MonetDB requires to compute the entire result. For example, interaction $V2$, creating the histogram over the attribute $education$, required 2.1s, far above the interactivity threshold.

In contrast, the traditional *Online Aggregation* and IDEA support approximate answers. For those two system, we report the latency to return an answer with less than a 1% standard error. That is, the IDEA system would even be able to return an earlier but less accurate answer.

As Figures 3 and 4 show, IDEA can almost always guarantee a response time below the 500ms threshold, while MonetDB and online aggregation frequently require several seconds.

The reasons are manifold: MonetDB is a column-store that heavily leverages both sorting and heavy compression to significantly improve query performance. In some cases, MonetDB can return results within the threshold; however, as the workflow progresses, queries begin to take several seconds because they involve compound predicates with several attributes. Similarly, our implementation of online aggregation can answer the simpler queries almost immediately, but has difficulty with more selective predicates as tuples satisfying the predicate become harder to find. To our surprise, MonetDB is sometimes faster than the online aggregation techniques (e.g., interaction S3). In most cases, the difference can be explained by the fact that MonetDB has the luxury of indexes and pre-processing time (e.g., to create more compact and compressed data), a luxury which would not exist within the *connect and explore* paradigm.

On the other hand, our A-WARE prototype augments basic online aggregation with additional optimizations. By intelligently sampling and caching (probability) results, our system is able to leverage past user interactions and immediately render some visualizations without rerunning the query (R1, S5, R2, V5, L7, L8). Our novel adaptive indexing techniques allow our system to scan only relevant subpopulations while saving space by indexing only rare tuples, which can substantially reduce latency for highly selective predicates (L3, L5, S7). Finally, since we reformulate the problem to use probabilities, our system can perform several query rewrites that are based upon probability theory, in order to to leverage past results and almost instantly return the result for a new query (S2, S3, S6, R3, N1). Through the combination of these novel techniques, our system almost always returns a result to the frontend within the interactivity threshold. Finally, the two interactions S4 and S7, which are above the interactivity threshold, could be brought below the threshold by using the previously mentioned prefetching strategies (which are not yet implemented) or tolerating a slightly higher error.

5. CONCLUSION

In this paper, we presented the case for Interactive Data Exploration Accelerators (IDEAs), which seek to maximize human productivity by allowing users to rapidly gain insights from new large datasets. We outlined some of the insights we gained from building one of the first IDEA systems and presented some initial results for an early prototype, called A-WARE, to show the advantages of our techniques compared to more traditional approaches.

6. ACKNOWLEDGMENTS

This research is funded in part by the Intel Science and Technology Center for Big Data, the NSF CAREER Award IIS-1453171, the Air Force YIP AWARD FA9550-15-1-0144, NSF IIS-1514491, and gifts from SAP, Oracle, Google, and Mellanox.

7. REFERENCES

[1] S. Agarwal et al. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *EuroSys*, pages 29–42, 2013.
[2] Apache Flink. http://flink.apache.org/.
[3] C. Binnig et al. The End of Slow Networks: It's Time for a Redesign. In *VLDB*, pages 528–539, 2016.
[4] C. Böhm, S. Berchtold, H. Kriegel, and U. Michel. Multidimensional Index Structures in Relational Databases. *J. Intell. Inf. Syst.*, pages 51–70, 2000.
[5] S. Chaudhuri, G. Das, and V. R. Narasayya. Optimized Stratified Sampling for Approximate Query Processing. *TODS*, 2007.
[6] A. Crotty et al. Vizdom Demo Video. https://vimeo.com/139165014.
[7] A. Crotty et al. An Architecture for Compiling UDF-centric Workflows. In *VLDB*, pages 1466–1477, 2015.
[8] A. Crotty et al. Vizdom: Interactive Analytics through Pen and Touch. In *VLDB*, pages 2024–2035, 2015.
[9] G. Cumming and S. Finch. Inference by Eye: Confidence Intervals and How to Read Pictures of Data. *American Psychologist*, pages 170–180, 2005.
[10] M. El-Hindi, Z. Zhao, C. Binnig, and T. Kraska. VisTrees: Fast Indexes for Interactive Data Exploration. In *HILDA*, 2016.
[11] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online Aggregation. In *SIGMOD*, pages 171–182, 1997.
[12] S. Idreos, M. L. Kersten, and S. Manegold. Database Cracking. In *CIDR*, pages 68–78, 2007.
[13] M. Lichman. UCI Machine Learning Repository, 2013.
[14] Z. Liu and J. Heer. The Effects of Interactive Latency on Exploratory Visual Analysis. *TVCG*, pages 2122–2131, 2014.
[15] Z. Liu, B. Jiang, and J. Heer. imMens: Real-time Visual Querying of Big Data. In *EuroVis*, pages 421–430, 2013.
[16] F. Olken and D. Rotem. Random Sampling from Relational Databases. In *VLDB*, pages 160–169, 1986.
[17] N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie. Online Aggregation for Large MapReduce Jobs. In *VLDB*, pages 1135–1145, 2011.
[18] The Apache Software Foundation. Hadoop. http://hadoop.apache.org.
[19] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized Streams: Fault-tolerant Streaming Computation at Scale. In *SOSP*, pages 423–438, 2013.
[20] M. Zaharia et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-memory Cluster Computing. In *NSDI*, pages 15–28, 2012.