SwiftTuna: Incrementally Exploring Large-scale Multidimensional Data

Jaemin Jo* Seoul National University

Wonjae Kim† Seoul National University

Seunghoon Yoo‡ Seoul National University

Bohyoung Kim§ Hankuk University of Foreign Studies

Jinwook Seo¶ Seoul National University

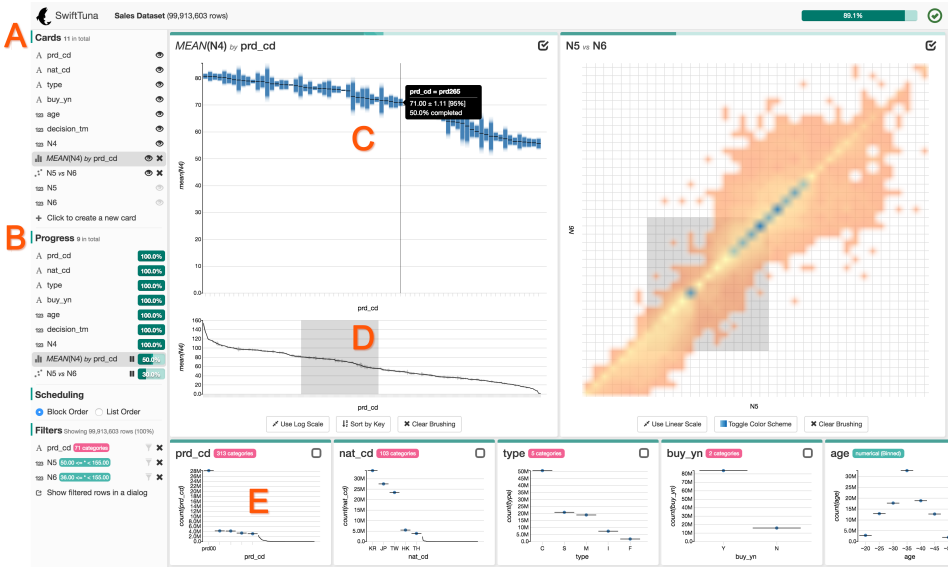

Figure 1: Interface of SwiftTuna. An analyst is exploring a multidimensional dataset with 100 millions of entries. Visualization cards (two expanded cards and five thumbnail cards) on the right side provide a univariate summary on a single dimension or visualize the relationship between two dimensions. The analyst expanded two visualization cards to further interact with them. The card list panel (A and B) on the left side shows the list of all visualization cards as well as the progress of each card. We used gradient plots (C) with a context view (D) to visualize uncertainty in incremental processing and tailed charts (E) to summarize a large number of categories on the x axis.

ABSTRACT

The advance in distributed computing technologies opens up new possibilities of data exploration even for datasets with a few billion entries. In this paper, we present SwiftTuna, an interactive system that brings in modern cluster computing technologies (i.e., in-memory computing) to InfoVis, allowing rapid and incremental exploration of large-scale multidimensional data without building precomputed data structures (e.g., data cubes). Our performance evaluation demonstrates that SwiftTuna enables data exploration of a real-world dataset with four billion records while preserving the latency between incremental responses within a few seconds.

Keywords: Large-scale data exploration, incremental querying.

1 INTRODUCTION

Although there have been great advances in visualization and database technologies, interactive visual analysis of large-scale multidimensional data is still challenging. The foremost issue is the long latency of queries that comes from the magnitude of the data. Querying on large-scale data often takes a few minutes or even a few hours, which not only delays decision making process but also makes users constantly pause analysis, hindering fluent exploration.

To resolve the latency issue, previous InfoVis system often adopted the preprocessing paradigm where a certain data structure (e.g., data cubes [1]) is precomputed from data in advance and used to answer future queries. Several InfoVis studies ameliorated data cubes to be more suitable for visual analytics: imMens [2] converted data cubes to multivariate data tiles to support interactive linking between visualizations, and Nanocubes [3] reduced the memory consumption of data cubes by sharing duplicate keys.

The preprocessing paradigm guarantees low latency for queries at the cost of time and space for preprocessing. For example, although Nanocubes slowed down the explosion of memory footprint of data cubes, it eventually reached the limitation of a single physical machine when the data became much larger [3].

We introduce SwiftTuna, a scalable approach to enable fluent visual exploration of large-scale multidimensional data. SwiftTuna does not reside on precomputed data structures. Rather, SwiftTuna incrementally processes visualization queries by splitting the raw data into blocks and processing each block on a computing cluster in a parallel and distributed manner. Thus, our approach allows queries that span multiple dimensions (e.g., filtering with multiple dimensions) without building large data structures. Also, even if the size of data exceeds the capability of a single machine, we can still handle the data by scaling the cluster's size, which was not possible in the previous studies [2][3].

*e-mail: jmjo@hcil.snu.ac.kr
†e-mail: wjkim@hcil.snu.ac.kr
‡e-mail: shyoo@hcil.snu.ac.kr
§e-mail: bkim@hufs.ac.kr
¶e-mail: jseo@snu.ac.kr

2 The SwiftTuna Design

SwiftTuna employs a client-server architecture. The client is a single-page web application where users can create queries, monitor the progress of the queries in real time, and interact with results to explore data (Figure 1). We present the design considerations and how we realized those considerations as follows:

DC1. Provide low-fidelity feedback promptly. A delayed response hinders users from observing the data and causes them to lose their attention. To enable fluid data exploration, we provide low-fidelity feedback promptly (i.e., prompt responses) based on a small sample from the data (i.e., 0.001% of entries).

DC2. Process results incrementally while estimating the final results. We visualize partial results of analytics and estimate the final results before a query is fully completed. We adopted gradient plots to visualize the uncertainty of partial results (i.e., 95% confidence intervals of counts and means, Figure 1C). This enables users to confirm or reject their hypotheses as early as possible during their exploratory analysis and thus test more hypotheses with limited time and resources.

DC3. Enable flexible scheduling. To amplify the use of partial results, the system provides flexible management of computing resources. For example, users can pause queries in real time if they think partial results are enough for decision-making. (Figure 1B)

DC4. Support multidimensional data exploration. SwiftTuna organizes multiple visualizations in a single view that show various aspects of data in a series of 1D or 2D projections including frequency histograms (for a categorical dimension), binned histograms (for a numerical dimension), pivot dot plots (aggregating a numerical dimension by a categorical dimension), and density plots (for two numerical dimensions) (Figure 1A).

DC5. Scalability in visualizations. To achieve scalability in visualizations, we adopted interaction techniques such as Focus+Context and brushing (Figure 1D). Also, to show categorical data in limited space, we designed a novel visualization, *tailed charts* (Figure 1E), which visualize prominent categories (e.g., the most frequent five categories) with salient visual elements in half of visual space, while the rest of the categories are outlined in another half of the space with a line (i.e., a tail).

On the server side, when a query from the client arrives (e.g., users applied a filter or added a new visualization), the server first processes the query only using a small sample for prompt responses. Then, the server equally splits the entire data into n blocks and processes the query using each block in a random order on an Apache Spark cluster. For example, suppose there is a query that calculates means of a numerical column (e.g., *age*) over a categorical column (e.g., *country*). For the first block, the cluster calculates the frequency, the sum of *age*, and the sum of squared *age* for a country. When finished, the server sends the result back to the client as an incremental result. The client accumulates incremental results using the column *country* as a key and visualizes the aggregated result with estimated confidence intervals. Then, the server moves on to the next block and this procedure is repeated until all blocks are processed or users stop the query.

3 Performance Benchmark

We conducted performance benchmarks using Criteo's Terabyte Click Logs dataset [4] that consisted of 4.3B entries with 40 dimensions. We created a cluster on Amazon Elastic Compute Cloud (EC2) with 16 spot instances of the r3.8xlarge tier. We measured 1) the latency of prompt responses (the time from when users requested a query to when the first feedback on the query was shown), 2) the mean interval between two successive incremental responses. We considered the number of blocks, n, as an independent factor and tested our system under two different values of n: 240 and 2,400 blocks.

Table 1. Results of Performance Benchmark

Type	Range or Cardinality	Prompt Responses (s)	Incremental Responses (s)[a]
Binned Histogram	0 – 35M	0.20±0.025	1.91±0.84 (3.54±1.58)
Density Plots	0 – 746K 0 – 35M	0.31±0.040	1.88±0.61 (3.46±1.05)
Frequency Histogram	20K	0.39±0.058	2.85±0.78 (3.93±1.31)
Pivot Dot Plots	7.4K	0.52±0.085	2.53±1.21 (3.88±0.93)

[a]Numbers are measurements with 2,400 blocks (and with 240 blocks).
Range or Cardinality: range (for a numerical dimension) or cardinality (for a categorical dimension) of dimensions related to each query.

Table 1 shows the results of the benchmark using four types of queries. Each block could be processed in three seconds for all types of queries, which means users could grasp updated results every three seconds. A smaller-size block (n = 2,400) yielded faster responses, as expected. However, a bigger block was preferred in terms of throughput. For example, when n was 2,400, it took 1.91 seconds to sweep 1.75 millions of rows in a block and create a binned histogram. However, it took only approximately two times longer (3.54 seconds) to process ten times more rows (17.5 millions of rows) when n was 240. This implies that an overhead such as network latency or the communication lag between workers lies in SwiftTuna which cannot be efficiently parallelized.

4 Research With SwiftTuna

Although SwiftTuna is still an ongoing work, we were able to demonstrate how modern distributed computing technologies are integrated into a visual analytics system to allow rapid and incremental data exploration without precomputation. We plan to extend the results of this research towards the following directions:

1. Providing a general platform for incremental visual analysis on clusters: We plan to combine each building blocks of SwiftTuna as a generalized platform so that analysts can easily extend the system for their own scenarios without editing or knowing all underlying architecture.
2. Designing scalable and incremental visualizations: As we designed tailed charts to visualize a number of categories, we seek novel visualization that are suitable for incremental and large-scale visualizations.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No. NRF-2014R1A2A2A03006998). This work was also supported by LINE Corporation.

References

[1] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology," *SIGMOD Record*, vol. 26, no. 1, pp. 65-74, Mar. 1997.

[2] Z. Liu, B. Jiang, and J. Heer, "ImMens: Real-time Visual Querying of Big Data," *Computer Graphics Forum*, vol. 32, no. 3, part. 4, pp. 421-430, 2013.

[3] L. Lins, J. T. Klosowski, and C. Scheidegger, "Nanocubes for Real-Time Exploration of Spatiotemporal Datasets," *IEEE Trans. Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2456-2465, Dec. 2013.

[4] Criteo's Terabyte Click Logs, http://labs.criteo.com/downloads/download-terabyte-click-logs, retrieved on March 2016.