A Benchmark for Interactive Data Cubes

Sean Stephens, Carlos Scheidegger

Abstract—The data cube is a convenient abstraction for exploring multi-dimensional datasets. However, exploring large datasets as data cubes at interactive speeds requires specialized data structures and systems. Such systems exist, but there is no comprehensive comparison of these systems on realistic data. We present a benchmark that uses real-world and synthetic spatiotemporal data to explore the performance of these systems on realistic query sets. We compare four existing systems using our implementation and demonstrate performance differences between them. Finally, we give a set of guidelines for choosing a system for applications.

Index Terms—Benchmark, Data Cube

1 Introduction

Data cubes are a useful tool for exploring aggregates of high dimensional data [**?**]. However, executing queries on data cubes of large datasets fast enough to support an interactive user interface is technically difficult because current systems use excessive memory usage or demand a high degree of parallelism [**?**, **?**].

To address these problems, systems such as *Nanocubes* and *Hashedcubes* allow fast queries in several dimensions by storing the dataset in a server-side structure on dedicated hardware [**?**][**?**]. This approach is much more complex and resources intensive than simpler approaches such as linear scans on a client, as in *datavore* and *crossfilter* [**?**, **?**]. On the other hand, it allows interactive exploration of data sets of tens of millions of points. [If we get imMens working we need to talk about that too.]

Each of these systems were published with performance information exercising the system on real-world test data. However, the published performance measures are not, in general, comparable. Different datasets were used, and critical parameters such as field resolution differ. As such, there is no complete picture of data/schema configurations each of these systems is best suited for, nor how each scales in different configurations.

This paper contributes a benchmark that allows direct comparisons of the query performance of these systems on realistic data. Additionally, we describe an implementation for *nanocubes* and *hashedcubes*. Finally, we give guidance for choosing between and configuring these systems based on dataset and application.

2 Related work

Large data visualization frequently requires computing aggregations over the entirety of a dataset to produce counts, histograms, and density plots [**?**]. For an interactive visualization, supporting arbitrary aggregations smoothly on large datasets presents a problem due to performance. To maintain an interactive user experience, a plot of aggregates should refresh with low latency (This is not merely an aesthetic concern; high latency induces large changes in a user's exploration to the detriment of analysis, as explored in [1] [Inline citation here?]). However, naive techniques for computing aggregations, namely linear scans over the dataset, do not scale to datasets with tens of millions of entries.

Several solutions exist for this problem. *Nanocubes* builds a hierarchical tree of agregations that fits in main memory, in contrast to traditional relational database data cube aggregations which must be stored on disk [**?**]. This system provides query performance that is fast enough for interactive applications, but is memory intensive. *hashedcubes* achieves similar performance for many common queries but consumes less memory by using a single sorted array, at the cost of poor performance in some cases.

[Talk about some/all of parallel systems, imMens, crossfilter, datavore].

1. mention streaming and why we are not covering it
2. Other benchmarks in databases, specifically on-disk cube (Not applicable because not specifically spatiotemporal, etc?)
3. "Implementing Data Cubes Efficiently" should be cited

3 Benchmark

1. Realistic benchmark given user sessions, best guess
2. Measuring: memory, query time, build time,
3. Which data sources and why
4. Which queries and why (synthetic and from user sessions)

4 Experiments

1. Which systems (nanocubes, hashedcubes, immens, crossfilter, datavore)
2. Plots, details of experiments (compiling, machine, how many times, more detail is better)
3. Interpretation of results

5 Discussion (of paper; what did we not do?)

1. how to improve this
2. what did we miss
3. Different interface/system induces different user behavior (cite Jeff Heer, The effects of interactive latency on explorative visual analysis)
4. Can be extended to new data, hopefully new queries

6 Conclusion and Future Work

1. What this means for making new systems (easier, negative space)
2. What next (making things parameterizable, what is the next research question)

Acknowledgments

The authors wish to thank A, B, C. This work was supported in part by a grant from XYZ.

References

[1] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2014.

• *Martha Stewart is with Martha Stewart Enterprises at Microsoft Research. E-mail: martha.stewart@marthastewart.com.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.
Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/