Extending BlinkDB to Support Matrix Operations

Nathan Harada, Julian Katz-Samuels

University of Michigan
{nharada, jkatzsam}@umich.edu

Abstract

We propose to extend BlinkDB to support matrix operations on large data sets. These operations will provide estimated error and confidence intervals similar to the current BlinkDB functions. We will develop the algorithms and commands, implement them in the current BlinkDB code, analyze their performance, and eventually merge our changes back into the open source project.

I. Introduction

As modern databases continue to grow in size, traditional database queries are beginning to take an unfeasibly long time to complete. Sampling databases, such as BlinkDB [1], attempt to offer meaningful results in a short period of time by providing an estimate of a query, along with a measure of the expected error. For staticians and data scientists, this "close enough" answer may be within acceptable error limits for data analytics and business intelligence. However, BlinkDB is currently limited to a few basic statistical operators (e.g., mean). We propose extending BlinkDB's functionality to include linear matrix operations. Matrix operations are essential for many statistical analysis techniques, such as regression or dimensionality reduction. By providing a "good enough" analysis of the data, we hope to reduce computing time required for such queries while maintaining meaningful accuracy. This work would enhance our ability to perform exploratory data analysis on large amounts of data.

Our proposal is three fold. First, we will develop approximation techniques for matrix operations (e.g., dot product, projection, dimensionality reduction) that are suitable for BlinkDB. That is, the techniques will be based on (i) how to create samples of the data offline and (ii) how to select a subset of these samples to approximate matrix operations. Second, we will implement these operations in BlinkDB, as well as extending the BlinkDB query language to support these operations. Last, we will test our results on both standard benchmarks and real world data, evaluating both performance and accuracy on systems of various sizes.

II. Related Work

Our project lies at the intersection of scientific databases and big data analytics. The SciBD project is relevant to our work because it is an analytical database that supports statistical and linear algebra operations [2]. Our project extends BlinkDB so that it can meet some of the same demands as SciDB. The advantage to extending BlinkDB is that because BlinkDB is a distributed sampling-based approximate query processing system, it holds the promise of executing matrix operations on large amounts of data at a much faster rate.

III. BlinkDB Extension

To add matrix operations to BlinkDB, we will need to both implement the underlying mathematical algorithms, as well as change the query language to support these operations.

I. Proposed Work

We propose introducing functions to both transform a selection of data into a matrix, as well as to run matrix operations on this data. The current SQL specification implemented in BlinkDB does not offer any way to specify matrix operations, thus we will have to create new schema to do so. For our purposes we will assume that we require all matrices to be stored in a the database and we will not accept them at query-time. One possibility is to nest functions and provide a function that refers to matrices, for example, to calculate the total value of an inventory. In addition, we will annotate matrix operations with an error bound or a time constraint. For example,

```
SELECT DOTPROD(VEC(prices), VEC(counts))
FROM inventory
WHERE prices > 10
WITHIN 5 SECONDS
```

for a time constraint or

```
SELECT DOTPROD(VEC(prices), VEC(counts))
FROM inventory
WHERE prices > 10
ERROR WITHIN 10% AT CONFIDENCE 95%
```

for an error constraint. The first query will return the total value of an inventory within 5 seconds with a 95% confidence interval. The second will do the same but instead of returning within 5 seconds will take however long required to execute the query with a maximum 10% error.

We will propose approximation techniques for fast processing of matrix operations. As in BlinkDB, there will be two stages: (i) offline sample creation and (ii) sample selection to approximate particular matrix operations. The availability of approximation techniques for matrix operations (e.g., projection or dimensionality reduction) that are appropriate for this framework will guide our choice of specific operations to implement.

II. Timeline

The initial stage of our project will concern itself with two goals. First, we will familiarize ourselves with the BlinkDB system so that we can extend it. We estimate that this will take a bout a month. Second, we will search the literature for approximation techniques for matrix operations that are suitable for the problem in question. The first part of this goal will be to determine the range of operations that are feasible to execute for BlinkDB. We expect that this substage will take around one to two weeks. The second part of this goal will be understand the relevant algorithms more deeply and figure out conceptually how to implement the algorithms. We expect this substage to take two to three weeks.

In the second stage of our project, we will implement the algorithms and run experiments. One key feature of this stage is that we hope to compare experimentally the suitability of various algorithms for use in BlinkDB. This will take the remainder of the semester.

IV. Testing

I. Proposed Work

To evaluate our extension, we will need testing data for our system. We plan to attempt to integrate SciDB's testing framework into our own benchmarks. SciDB is a now commercial database that bases its underlying structure on matricies, and thus requires matrix operations for test. We also plan to develop our own testing metrics based on expected use cases.

For actual testing we will run BlinkDB in both standalone and distributed mode. BlinkDB is designed to excel at combing through large data quickly, and thus we will need a large database to test our implementation. We plan to use Amazon Web Services to provide cluster computing nodes for our test. Using a cluster of 10 t2.medium instances for 24 hours of computation will cost approximately $12.50, not including EBS costs. We think this cost is within an acceptable personal budget, but if we require additional nodes or

compute time we will attempt to pursue funding via the university if possible.

II. Potential Roadblocks

We expect the majority of testing roadblocks will come from the complexity of deploying BlinkDB on a cluster of machines. Neither of us have significant experience with distributed cloud computing, although we have worked with Amazon Web Services before. Additionally, because we are developing a system for linear matrix operators, traditional test benchmarks like TPC-H are likely not usable in our case. Because SciDB is now a proprietry system it may also be difficult to aquire any testing insights or data they may use.

V. Conclusion

By extending BlinkDB to include linear matrix operators we hope to allow users to generate more expressive queries in a shorter time period on large datasets. Current aggregate functions offer a limited look at the data, especially in cases where the user is attempting to apply statistical techniques to a database. Additionally, we are excited to give back to the open source community, and hope to merge our changes in with the main BlinkDB repository after the project is complete.

References

[1] Agarwal, Sameer, et al. "BlinkDB: queries with bounded errors and bounded response times on very large data." Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013.

[2] Brown, Paul G. "Overview of SciDB: large scale array storage, processing and analysis." Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010.