BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data

Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, Ion Stoica

Paper Summary by Group 5:
Alok Thatikunta (111498776), Rahul Sihag (111462160), Sweta Kumari (111497926)
{athatikunta, rsihag, swkumari} @cs.stonybrook.edu

BlinkDB is an approximate query engine for interactive SQL queries on big data. Running SQL queries on terabytes of data can take lots of time. For many applications, we want to know the answer to a query in real time. Answering these queries can be done if we were to execute them on fewer rows. BlinkDB uses sampling to determine the result of a query and even provides error bounds on the answer.

BlinkDB extends the Hive framework. It adds an offline sampling module and a runtime sample selection module. SQL queries can be annotated with an error or a time constraint. Based on the given constraints, the appropriate sample size is selected.

An optimization model can range from a simple model which knows most about the future queries or nothing about the queries in advance. If one knows the future queries then we can optimize our database for such queries resulting in maximum efficiency, but very low flexibility when an unknown query is given. If based on past data, we are able to make inferences about the query predicates, this results in slightly increased flexibility and relatively lower efficiency compared to previous. Next, if we can make a model which estimates only based on the columns which are being queried upon, we can build indices on these and optimize data access. These sets of columns which are selected are known as query column sets. In the extreme of the spectrum, we have a model which assumes that queries are unpredictable. BlinkDB uses a model of predictable query column sets.

Sample creation module creates stratified samples on the most frequently used Query Column sets to ensure efficient execution for queries on rare values. BlinkDB performs multi-dimensional stratified sampling. As performing sampling on the entire table has huge storage costs, few subsets of columns are selected on which to samples are built. This is framed as an optimization problem which takes parameters Sparsity of data, Workload and Storage cost associated with selecting a column. Goal function tries to maximize the total columns selected subject to the condition that storage cost of selected columns fits in the storage budget.

To answer a query, query columns which are a subset of the stratified column are selected. If such columns do not exist then the query is executed on all samples maintained and samples are selected based on selectivity towards the query. Next, BlinkDB selects appropriately sized subsamples based on query response time or error constraints provided. Error latency profile is built by running the query on smaller samples to estimate selectivity and project latency, error for larger samples. Error profile is created for queries with error constraints and a latency profile for time constraints.

Performance of BlinkDB is compared on data from Conviva and TPC-H benchmark. It is shown that by slightly trading off the accuracy of the final answer, there is an order of magnitude

improvement in query response time. They compare the average error per query column set against uniform sampling, single-dimensional stratified sampling, and multi-dimensional stratified sampling. It is observed for common query columns sets, the multi-dimensional method produces smaller statistical errors. Finally, a comparison is also done on the effectiveness at meeting the time or error bounds requested by the user. It is shown that BlinkDB is able to accurately select samples to satisfy target response time.