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

**Introduction:** Modern data analytics applications usually involve computing aggregates over a large number of records. Traditionally, such queries have been executed using sequential scans over a large fraction of a database. But increasingly, some applications require near real-time response rates. In such cases, the applications can usually tolerate certain amount of error, but the requires a fast response. BlinkDB is a massively parallel approximate query engine for running interactive SQL queries on large volumes of data which provides fast response rate with bounded errors. It allows the user to balance response rate with response time flexibly.

**Literature Review:** One of the most important aspect that any sampling based query processor needs to take account of is the future workload assumption while creating workload model. There are basically four main approaches namely predictable queries, predictable query predicates, predictable QCs and unpredictable queries, where predictable queries provides lowest flexibility but guarantees high efficiency. Also, QCs models are relatively more stable over time, that is, it makes good use of past queries to predict future workload.

**Implementation:** BlinkDB consists of two main modules, 1) Sample Creation and 2) Sample Selection. Sample creation module creates stratified samples on the most frequently used QCSs(query column sets, the columns used by WHERE, GROUP BY, and HAVING) to ensure efficient execution for queries on rare values. Sample creation is formulated into an optimization problem. BlinkDB choose a collection of stratified samples with total storage costs below some user configurable storage threshold which are neither over- nor under -specialized for the query workload.

BlinkDB extends Hive framework by adding two major components, 1) an offline sampling module that creates and maintains samples over time, and 2) a run-time sample selection module that creates an *Error-Latency Profile*(ELP) for queries. Samples are decided by QCSs that appear in queries, once the QCSs is decided, BlinkDB do distributed reservoir sampling or binomial sampling techniques to create a range of uniform and stratified samples samples across a number of dimensions. At run-time, ELP decides the sample to run the query after considering the error and response time requirements.

**Results:** The paper presents a new, working database system capable of running SQL queries in an approximate fashion. Results show that the authors achieved a 10 to 200x speedup for queries with a 1% error bound at 95% confidence, compared to standard Hive on Spark (both with and without caching), due to having to read far fewer data points. As the authors note, in some cases, BlinkDB returned results in a few seconds while it took thousands of seconds for the other systems. This was true for both the TPC-H benchmark and Conviva, which is more representative of a real-world workload. Additionally, the authors’ decision to utilize stratified sampling resulted in lower error compared to uniform sampling.

**Discussion:** The main disadvantage of the paper is that BlinkDB only focuses on aggregate queries. However, non-aggregate queries are also important to enterprises, because for example, we can do personal recommendations based on each user’s view history. But this model doesn’t support non-aggregate queries, which can be further explored.Secondly, the author mentioned that unique QCs are tested for evaluation on different sampling strategies but didn’t mention how these QCs are selected. Another disadvantage of BlinkDB is that it does not support arbitrary joins and nested SQL queries.

**Conclusion:** This work will be beneficial in the coming years as it provides a flexible mechanism to balance response time and error rate which is needed in the big data systems.