Scriber Slides

Asad Imtiaz Butt

Summary

- Motivation
  - Disk reads are expensive, so achieve low latency by maximizing the amount of data in memory
- Key Contribution
  - Compressed representation of the input data
  - New algorithm for querying compressed data
- Evaluation
  - Able to fit 242 GB of data on a cluster with a total memory of 150 GB, whereas Cassandra and Mongo can fit a maximum of 23 GB

Pros

1. Low storage - does not use secondary indexes
2. Low latency - in-memory data access are much faster
3. Higher throughput than other systems if indexes do not fit in memory
4. Their query algorithm is 2.3x faster on average in comparison to strawman

Cons

1. Expensive preprocessing step (4GB/hour/core)
2. Updates cannot be in-place
3. Poor throughput for queries involving sequential scans
4. Does not provide fault tolerance
5. Experiments designed arounds strengths of Succinct rather than representative of real world scenarios

Discussion

1. How would Succinct perform if the input size became so large that it cannot be stored just in memory?
2. Can we create a similar system for structured data?
3. Is there a way to bring together BlinkDB and Succinct?
4. What kind of workloads would Succinct not be useful for, given lack of support for in-place updates?
5. What if we have binary data instead of text files?