Querying Large Scientific Databases

Luciano and Brian
Scientific Data challenges

Data is in the Terabytes

Queries are exploratory

“Paraphrased, the answer is in the database, but the Nobel-prize winning query is still unknown”

Unhelpful query results that take minutes to process

Q: select * from PhotoObj. 4 TB Database

VS

number of people learning english

Show results for number of people learning english

Search instead for number of people learning english

Answer: It is estimated that over 1 billion people are currently learning English worldwide. According to the British Council, as of the year 2000 there were 150 million English as a Foreign language speakers. In addition, there were 375 million English as a Second Language speakers.

ESL Market Statistics: How Many People Learn English?
est/about.com/englishteachingresources/efl_earnermarket.htm
Research Directions

One-minute database kernels for real-time performance.

Multi-scale query processing for gradual exploration.

Result-set post processing for conveying meaningful data.

SELECT COUNT(*)
FROM Sessions
WHERE Genre = 'western'
GROUP BY OS
WITHIN 5 SECONDS
Research Directions

Figure 1: An example of query morphing.

Query morphing to adjust for proximity results.

Query alternatives to cope with lack of providence.

---

Q: select * from PhotoObj.

-- Q1: Using the time budget. (36291322 tuples)
SELECT ra, dec, band1, intensity1, type
FROM PhotoObj;

-- Q2: Using data statistics. (879300 tuples)
SELECT * FROM PhotoObj
WHERE ra BETWEEN 53 AND 54
AND dec BETWEEN 80 AND 82;

-- Q3: Using query statistics. (899 tuples)
SELECT * FROM PhotoObj
WHERE ra BETWEEN 53 AND 54
AND dec BETWEEN 80 AND 82
AND distance(ra,dec,radius) < 10;
An Introduction to BlinkDB
Motivation of BlinkDB

Real time analytics for web companies

- 4 BILLION + Streams per Month
- 1.6 BILLION + Devices per Month
- 400+ Premium Media Video Players
- 180+ Countries
Taxonomy of Solutions

Low flexibility / High Efficiency
- Predictable Queries

Predictable Query Predicates

Predictable Query Column Sets

High flexibility / Low Efficiency
- Unpredictable Queries

Flexibility Efficiency
Query Column Sets - Include Table

```sql
SELECT country, SUM(clicks), AVG(time_on_page), FROM website_data WHERE referral_website = 'Google' GROUPBY country
```

Query Column Sets consist of the columns in the WHERE, HAVING, and GROUPBY fields but not in the payload columns.

Query Column Set - referral_website, country
The most popular queries use very few Query Column Sets

The Query Column Sets being used are stable over time
Limitations: Lack of (Full) Support for Joins

Full support only for joins where one table fits entirely into memory.

How much is this a problem?
System Overview

Integration into Hive

Modifications to HiveQL to support confidence intervals, error estimates, and time constraints.

Integration of sample selection into plan formation.

Creation and maintenance of samples. Integrates into the distributed file system and cache.

Run query Q over file cache when deciding on sample selection.
HiveQL Extension

SELECT COUNT(*)
FROM Sessions
WHERE Genre = 'western'
GROUP BY OS

ERROR WITHIN 10% AT CONFIDENCE 95%
Stratified Samples

Uniform Sample

Query
Stratified Sampling (Continued)
Storage of Stratified Samples
Limitations of Stratified Samples

Large

Expensive to create

Sorting
How to Create Samples

- Number of occurrences of each possible $|\phi|$-tuple
- Sort the table by the columns in $\phi$
- For each $|\phi|$-tuple that has a number of occurrences greater than $K$, we need to randomly sample from those occurrences
- Finally, we write the sample out to disk

How long does this take?

According to authors, 5-30 minutes. (To be examined later)
What Samples should we create?

We want column query sets that are used by our workload.

Want to use QCS that will take up less space.
Sample Selection and Runtime
Sample Selection and Runtime

Query Planner

QCS1

QCS2

Stratified Sample

Uniform Sample
Sample Selection and Runtime

Figure 6. Error Latency Profiles for a variety of samples when executing a query to calculate average session time in Galena. (a) Shows the ELP for a sample biased on date and country, (b) is the ELP for a sample biased on date and designated media area (dma), and (c) is the ELP for a sample biased on date and the ended_flag.
Sample Selection and Runtime
Results

(a) Biased Samples (Conviva)

(b) Biased Samples (TPC-H)
90% of the queries come from 10% of the templates. However, this is still 18 Query Column Sets for Conviva and 45 different Query Column Sets for Facebook.
Performance - Simple Query on Filter, Groupby, Avg.

(c) BlinkDB Vs. No Sampling
Results

- Is this experiment fair? What portion of the speedup is due to the probabilistic framework? What portion is due to the clustering & redundancy that BlinkDB employs?

- BlinkDB states their sample build time should be between 5-30 minutes. This experiment shows a simple two column filter, groupby, aggregate query to take 20 minutes. Does this seem reasonable?
Accuracy of Results

(a) Error Comparison (Conviva)
(b) Error Comparison (TPC-H)
(c) Error Convergence (Conviva)

(a) Response Time Bounds
(b) Relative Error Bounds
(c) Scaleup
Extensions and Open Questions

1) Could we carry around only certain payload columns with each QCS?

2) Could we build this index partially and incrementally? Why or why not?

3) How does this apply to scientific data management?