Online aggregation & Sampling from Joins

CompSci 590.02
Instructor: Ashwin Machanavajjhala
Outline

• Online Aggregation

• Ripple Joins

• On the hardness of sampling from Joins
Online Aggregation

• Most systems compute aggregated like averages/counts/etc. exactly.

• But aggregates only provide a “summary-view” of the data.

• Why wait for an aggregate computation on the entire data?
Online Aggregation

AVG: 2.6336
Confidence: 95
Interval: 0.0652539

14% done
Examples of Queries

• Select Sum(Salary) From R

DISTINCT

• Select Count(DISTINCT hashtags) from T

GroupBy

• Select Average(Grade) from STable GroupBy CourseID

JOIN

• Select Sum(Grade*Difficulty) from STable, Course
Example Scenarios

• Compute the number of individuals in the table that satisfy function F, where F is a computationally intensive property.
  – Running the query on the entire data takes $O(nf)$, where $f$ is the time for checking F on one record.

  – We can get an approximate answer much faster ...
Example Scenarios

• Compute the sum of all elements in a database, which is partitioned on k machines.
  – Compute sum on each machine $S_i$, and then add up all the $S_i$’s
  – Time taken to compute aggregate $= \max(\text{time taken by one machine})$
  – If a machine fails ...
Example Scenarios

• Find the number of people in database D1 also appears in database D2
  – Exact answer needs checking $|D1| \cdot |D2|$ pairs of records.
  – Can we get an approximate answer faster?
Aggregations on a single table

1. Read the records of the table in a random order

2. Maintain a *running estimate* of the required aggregate

3. Compute confidence bounds on the error in the running estimate.
Random access

• Random I/Os are expensive

• Heap Scans
  – Heaps maintain the data in the order in which they are inserted
  – If insertion order is not correlated with values, then this can be used instead of a true random ordering

• Index Scans
  – If index is on an attribute that is not the same as the aggregated column

• Sampling from indexes
  – From previous class
Group-By

• E.g., Select \( \text{Avg}(\text{Salary}) \) from \( R \) GroupBy Department

• Standard technique
  – Sort the relation by the grouping attribute
  – Compute the within group aggregate by scanning the sorted output

• Sorting is a blocking operation

• Alternative : Hashing
Running Estimate

• If N is the number of tuples in the data
• If n is the number of tuples seen ...

• SUM : $N/n$ (current sum)
• COUNT: $N/n$ (current count)
• AVG : $1/n$ (current sum)
Confidence bounds

Assuming the input tuples are randomly chosen. If $X_i$ is the random variable corresponding to the $i^{th}$ tuple, then $X_1, X_2, \ldots$ are independent random variables.

$$P\{|Y_n - \mu| > \varepsilon\} < 2 \exp\{-2n\varepsilon^2 / (b-a)^2\}$$

Where

- $Y_n$ is the running estimate after seeing $n$ elements
- $\mu$ is the actual aggregate
- $[a,b]$: range of the values in the database
Online Aggregation over Joins

• How to generate a random ordering of pairs of tuples from the Join of a relation?

  – Option 1: Compute the join and then read the output of the join in a random order – BLOCKING!

  – Option 2: Nested Loop Join (over random orderings of the two tables)
Nested Loop Join

Inner Relation

Outer Relation
Nested Loop Join

Inner Relation

Unnecessary work is done if:
- Values in the inner relation are roughly the same
- Output of the aggregate is not very sensitive to the values in the inner relation
Ripple Join

Read x records from each table, and compute the join on these records.
Ripple Join

Inner Relation

Outer Relation

Read x records from each table, and compute the join on these records.
Ripple Join

Read $x$ records from each table, and compute the join on these records.
Online aggregation with Joins

• The output tuples are no longer independent samples from the underlying distribution
  – Why?
Difficulty of Join Sampling

• Sample(Join(R,S)) \neq Join(Sample(R), Sample(S))

• R: \{(a, x0), (b, x1), (b,x2), ..., (b,xn)\}
• S: \{(b,y0), (a,y1), (a,y2), ..., (a,yn)\}

• In R \times S: Half the records have ‘a’ and half the records have ‘b’

• In Sample(R): probability ‘a’ appears is very small.
Using statistics

• If we know for each tuple \( t \in R \), how many tuples it joins with in \( S \) (call it \( n_s(t) \))

• Pick a random tuple \( t \in R \)
• Include it with probability proportional to \( n_s(t) \)
Summary

• Online aggregation helps provide approximate answers without waiting for the exact answer

• Requires iterating over a random order of the data

• Sampling over Joins is difficult.