Query Processing, Approximation, and Resource Management in a Data Stream Management System

Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Manku, Chris Olston, Justin Rosenstein, Rohit Varma

Presentation by:

Abram Hindle

Department of Computer Science

University of Waterloo

ahindle@cs.uwaterloo.ca

October 4, 2005

This Presentation

- What am I going to cover?
  - Authors
  - Introduction
  - DSMS
  - STREAM
  - Summary

Authors

- Most authors come from the Stanford DB group
- Rajeev Motwani - Professor (CS) and Director of Grad Studies at Stanford
- Jennifer Widom - Professor (CS,EE) at Stanford
- Arvind Arasu - Ph D. Candidate (CS) at Stanford (Widom)
- Brian Babcock - Ph D. Candidate (CS) at Stanford (Motwani)
- Shivnath Babu - Ph D. Candidate (CS) at Stanford (Motwani and Widom)
- Mayur Datar - Ph D.(Motwani)
- Gurmeet Manku - Ph D. - Works for Google now
- Christopher Olston - Assistant Professor (CS) at Carnegie Mellon
- Justin Rosenstein - Msc (CS) - Works for Google now
- Rohit Varma - Standford Student?

Introduction

- STREAM (Stanford Stream Manager) is a Data Stream Management System (DSMS) from Stanford
- SQL Extension for Streams
- Query Plans, Approximation, etc
- General overview of a prototypical DSMS

DSMS

- Data Stream Management Systems
- Like RDBMS but allows streams
- Streams consist of sequential, temporal data,
- Stream can be continuous, time varying, unbounded
- Queries can be streams
- New data comes in at various intervals
- A DSMS can query and help reason about data streams.
- Like a pipe and filter architecture for DBMS
  - Primary difference: DSMS are for querying where was dataflow/pipe and filter architectures might not be used for querying.

DSMS Challenges

- Changing conditions of streams
- Query Accuracy
- Run Time Conditions
- Data Rates
- Query Loads
- Windowing, Queuing, Aggregation

Context

- DSMS implementations existed at the time
  - Aurora
  - Hancock w/ AT&T
  - TelegraphCQ and NiagraCQ

Contributions

- SQL extension
- Plan Sharing and approximating Query Plans
- Static and Dynamic Approximation
- Algorithms for resource allocations
- Constraints based resource allocation
- Scheduling Algorithm
- Implementation

Query Language Extension

- An extension of SQL (SELECT, CREATE STREAM)
- Streams can be built up from other continuous queries
- `FROM` is extended to reference streams and the windowing to use
  - `RANGE` for time intervals (`1 DAY PRECEDING`)
  - `ROWS` selects preceding rows
  - `WHERE` clause is optional
  - `SAMPLE` selects a (uniform?) sampling
  - `PARTITION BY` is a group by for streams

Query Language Example

- Select from a web proxy
- Number of requests during the last day

```
SELECT COUNT(*)
FROM   Requests S [RANGE 1 DAY PRECEDING]
WHERE S.domain = ‘stanford.edu’
```


Query Language Example

- Select from a web proxy with partitions
- How many requests, considering only a max of 10 per client

```
SELECT COUNT(*)
FROM   Requests S
 [PARTITION BY S.client id
  ROWS 10 PRECEDING
  WHERE S.domain = ‘stanford.edu’]
WHERE S.URL LIKE ‘http://cs.stanford.edu/%’
```


Query Language Example

- Sub select with Sampling
- Sample 10% of 10000 preceding rows and count how many are commerce

```
SELECT COUNT(*)
FROM
  (SELECT R.class
   FROM Requests S 10% SAMPLE, Domains R
   WHERE S.domain = R.domain) T
  [ROWS 10000 PRECEDING]
WHERE T.class = ‘commerce’
```

Formal Semantics

- $C(S)$ - current timestamp of stream S
  - Explicit Timestamps $C(S) = \tau' - B$ (B scrambling bound)
  - Arrival Timestamps $C(S)$ is the system clock
  - $C(Q)$ for query Q is the minimum $C(S)$ for streams referenced by Q
- $A(Q, \tau)$ is the answer of query Q at time τ
  - Continuous Queries: $A(Q, C(Q))$
  - Streamed Queries: $\cup_{\tau \leq C(Q)} A(Q, \tau)$ (only the smallest τ are in result)

Query Plans

- STREAM uses somewhat shared query plans but not a global plan like Aurora.
- Plan consists of
  - Query operators - operators read the tuples from the stream and write output to queues
  - Inter-operator queries - operators connected by queues
  - Synopses - tuple summary (aggregates, over whole history or windows)

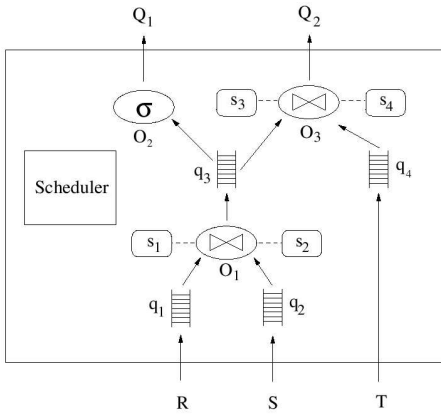

Figure 1: Plans for queries Q_1,Q_2 over streams R,S,T.

Dataflow

- Variable Rate Flows
- Issues of Shared Data Plans
  - Variable Consumption Rate (shared queues not a good idea then)
  - Approximation with all the queue mayhem, approximation might be best
  - Synopses sharing
- This done heavily in Computer Music, Computer Audio, Digital Filters, and just about any data flow application, what was shown here is not very new at all.

Static Approximation

- Window Reduction
  - reduce window sizes
- Sampling Rate Reduction
  - decrease sampling rate based on the requirements of the query.

Dynamic Approximation

- Assumptions: Less Memory Overhead is better performance
- Synopsis Compression
  - Reduce Memory Overhead
- Sampling and Load Shedding
  - Aggregate or throw out tuples.
  - Add more sampling (reduce storage)
  - There are a few clever *randomized* algorithms that allow you to keep a uniformly distributed sample

Resource Management

- Memory, CPU, I/O, Network
- More Implementation Based
  - "Maximize" result precision through statically allocating memory for a query. Algorithm: assume you know just about everything about the resource usage, then use a numerical solver to find the best precision solution.
  - Incorporate known constraints to reduce synopsis sizes. Assume based on prior knowledge the behavior of the system how data arrives, (e.g. in clusters around OrderID), use this information in joins.
  - Scheduling, either tuples processed as they arrive or process n tuples and push the results through the system. (This has been studied to death in the Computer Music field and Operating Systems)
- "provably has close-to-optimal queue size overhead"

Their Conclusions

- SQL Extension for DSMS
- Query Plans w/ Approximation
- Static and Dynamic Approximations
- Resource allocation algorithms

Issues

- Overview paper
- Naive Algorithms used in other fields
- Ignoring work done in non-DB fields
- Ignoring Temporal DBMS (TQuel, ATSQL)

Summary

- This paper was an overview of STREAM (a DSMS)
- The largest contribution was their SQL Extension to streams
- Provided details regarding data flow issues

Questions

- What other research could they have looked at?
- What other fields have similar problems with Stream data?
- What are lessons from other fields?
- Is their SQL extension the right route? What would you add change or remove?