STREAM
Stanford Data Stream Management System

Pranav Moktali
STREAM
Stanford Data Stream Management System

Pranav Moktali
LAST YEAR WE RECOGNIZED THAT OUR PROCESSES WERE FAR TOO COMPLEX.

SO WE PUT THEM INTO THE CLOUD.

LET THE CLOUDS MAKE YOUR LIFE EASIER.
Learnings so far...
Hive

- Abstraction over Hadoop.
- Supports SQL-like queries.
- Runs Map-Reduces jobs from declarative statements.

Here comes continuous stream of data!
Apache Storm

- Stream processing system
- Spouts and Bolt
- A record is a tuple of <key, value(s)>

What about to SQL-like query?
Stanford Data Stream Management System

- Project from Academia - Stanford (2003-2006)
- Different approach for Stream processing
- Prototype - DSMS
Data Stream Management System

- CQL: Continuous Query Language
- CQL Under the hood
- Optimize
- Handle high volume
- User Interface to query and visualize
Continuous Query Language

Abstract semantics for continuous queries

Diagram:

- relation-to-stream
  - QStream - query stream
  - RStream - result stream
- relation-to-relation
- stream-to-relation
  - tuple-based sliding window
  - time-based sliding window
  - partitioned sliding window

Examples:
- SQL-based expressions
- Use SQL constructs in expressions
• Unbound bag (multiset) of pairs $<s, t>$
• $s$ is a tuple in stream $S$
• $t$ is a timestamp
• $t$ denotes logical time of arrival of $s$
• Leslie Lamport: happened-before
• $R$ is a time-varying bag of tuple
• $R(t)$ denotes bag of tuples at time $t$
Continuous Query Language

Abstract semantics for continuous queries

- *relation-to-stream*
  - Join: join stream
  - Filter: filter stream
  - Project: project stream
- *relation-to-relation*
  - SQL constructs in expressions
- *stream-to-relation*
  - Tuple-based sliding window
  - Event-based sliding window
  - Partitioned sliding window

Examples:

- Select language from a given list: example list
- Aggregate: mean, sum, min, max
  - filters: if, when
  - window functions: row_number, rank, dense_rank
Uses SQL construct to expressions
- tuple-based sliding window
- time-based sliding window
- partitioned sliding window

stream-to-relation
tuple-based

- Input:
  - int $N > 0$ at $t$
- Output:
  - $R(t)$, $N$ tuples with timestamp $\leq t$
- Usage
  - [Rows $N$]
  - [Rows Unbounded]
time-based

- Input:
  - timestamp \( w \)
- Output:
  - \( R(t) \), tuples between \( t-w \) and \( t \)
  - [Range \( w \)]
  - [Now] (\( w=0 \))
partitioned

• Input:
  • int $N$ and set $\{A_1, \ldots, A_k\}$

• Output:
  • $R(t)$, union of substreams by attr of size $N$

  • [Partition By $A_1, \ldots, A_k$ Rows $N$]
relation-to-stream

- \textit{Istream} - insert stream
- \textit{Dstream} - delete stream
- \textit{Rstream} - relation stream
Continuous Query Language

Abstract semantics for continuous queries

Diagram:

- relation-to-stream
  - DeltaStream - delta stream
  - FoldStream - fold stream
  - FlowStream - flow stream

- stream-to-relation
  - Tuple-based sliding window
  - Event-based sliding window
  - Partitioned sliding window

- relation-to-relation
  - Uses SQL constructs in expressions

Examples

- [Example 1]
- [Example 2]
- [Example 3]
Examples

Select Istream(*) From S [Rows Unbounded] Where S.A > 10

Select * From S1 [Rows 1000], S2 [Range 2 Minutes]
Where S1.A = S2.A And S1.A > 10
Istream(*)
r-to-s
[Rows Unbounded] s-to-r
S.A > 10

r-to-r
Examples

Select Istream(*) From S [Rows Unbounded] Where S.A > 10

Select * From S1 [Rows 1000], S2 [Range 2 Minutes] Where S1.A = S2.A And S1.A > 10
Query Plans and Execution

- CQL to a Query Plan
  - Operators
  - Queues
  - Synopsis
- Example Query Plan
<table>
<thead>
<tr>
<th>Name</th>
<th>Operator Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>select</td>
<td>relation-to-relation</td>
<td>Filters elements based on predicate(s)</td>
</tr>
<tr>
<td>project</td>
<td>relation-to-relation</td>
<td>Duplicate-preserving projection</td>
</tr>
<tr>
<td>binary-join</td>
<td>relation-to-relation</td>
<td>Joins two input relations</td>
</tr>
<tr>
<td>mjoin</td>
<td>relation-to-relation</td>
<td>Multiway join from [22]</td>
</tr>
<tr>
<td>union</td>
<td>relation-to-relation</td>
<td>Bag union</td>
</tr>
<tr>
<td>except</td>
<td>relation-to-relation</td>
<td>Bag difference</td>
</tr>
<tr>
<td>intersect</td>
<td>relation-to-relation</td>
<td>Bag intersection</td>
</tr>
<tr>
<td>antisemijoin</td>
<td>relation-to-relation</td>
<td>Antisemijoin of two input relations</td>
</tr>
<tr>
<td>aggregate</td>
<td>relation-to-relation</td>
<td>Performs grouping and aggregation</td>
</tr>
<tr>
<td>duplicate-eliminate</td>
<td>relation-to-relation</td>
<td>Performs duplicate elimination</td>
</tr>
<tr>
<td>seq-window</td>
<td>stream-to-relation</td>
<td>Implements time-based, tuple-based, and partitioned windows</td>
</tr>
<tr>
<td>i-stream</td>
<td>relation-to-stream</td>
<td>Implements Istream semantics</td>
</tr>
<tr>
<td>d-stream</td>
<td>relation-to-stream</td>
<td>Implements Dstream semantics</td>
</tr>
<tr>
<td>r-stream</td>
<td>relation-to-stream</td>
<td>Implements Rstream semantics</td>
</tr>
</tbody>
</table>

**Table 1.** Operators used in STREAM query plans.
Query Plans and Execution

- CQL to a Query Plan
  - Operators
  - Queues
  - Synopsis
- Example Query Plan
Select * From S1 [Rows 1000], S2 [Range 2 Minutes] Where S1.A = S2.A And S1.A > 10
Performance Issue

- Making it more efficient by removing redundancy
  - Synopsis Sharing
  - Exploiting Constraints
  - Operator Scheduling
Fig. 3. A query plan illustrating synopsis sharing.
Exploiting Constraints

- k-constraint
  - referential integrity
  - ordered-arrival
  - clustered-arrival
Operator Scheduling

Possible schedulers
- FIFO
- Greedy

Introducing Chain Scheduling
- Greedy chain creation
- FIFO operations in chain
Adaptivity

Query load varies - StreaMon

Fig. 4. Adaptive query processing.

(a) StreaMon
(b) k-Mon
(c) A-Greedy
Approximation

Handling the peaks in data stream
  • CPU-Limited approximation
    • When CPU time is insufficient to handle data
  • Memory Limited Approximation
    • When data stream exceeds memory capacity
Discussion

- Distributed processing *(Hadoop, Cassandra)*
- Crash Recovery *(Storm)*
  - No ACID
  - Need "catch-up" or eventual consistency.
- Improved Approximation
- Relationship to Publish-Subscribe Systems

- SQL-like interface
- Chain Scheduling performance
  - need experimental results
- Are synopsis overhead?
- UI
- Did the authors compare STREAM with any other systems?
STREAM
Stanford Data Stream Management System

Pranav Moktali