Streaming Data Integration: Challenges and Opportunities

Nesime Tatbul
Talk Outline

• Integrated data stream processing
• An example project: MaxStream
  – Architecture
  – Query model
• Conclusions
Data Stream Processing

- Monitoring applications require collecting, processing, disseminating, and reacting to real-time events from push-based data sources.
- “Store and Pull” model of traditional databases does not work well.

[Diagram showing the difference between Traditional Database Systems and Stream Processing Engines]

ICDE NTII Workshop, 2010
Nesime Tatbul, ETH Zurich
“Integrated” Data Stream Processing

• Today, integration support for SPEs is needed in three main forms:
  1. across multiple streaming data sources
     • example: news feeds, weather sensors, traffic cameras
  2. over multiple SPEs
     • example: supply-chain management
  3. between SPEs and traditional DBMSs
     • example: operational business intelligence
#1: Streaming Data Source Integration

- **Goal:** Integrated querying over multiple, potentially heterogeneous streaming data sources
- **Challenges:**
  - Schemas of different sources can differ from one another and from the input schemas of the already running CQs.
  - Input sources or the network can introduce imperfections into the stream.
  - Adapters may become a bottleneck.
- **Current state of the art:**
  - Commercial SPEs offer a collection of common adapters and SDKs for developing custom ones.
  - Mapping Data to Queries [Hentschel et al]
  - ASPEN project [Ives et al]
#2: SPE-SPE Integration

**Goal:** Integrated querying over multiple, potentially heterogeneous SPEs
- to exploit the advantages of distributed operation
- to exploit specialized capabilities and strengths of SPEs
- to provide higher-level monitoring over large-scale enterprises with loosely-coupled operational units

**Challenges:**
- The need for functional integration
- The need to deal with heterogeneity at different levels (e.g., query models, capabilities, performance, interfaces)

**Current state of the art:**
- MaxStream project [Tatbul et al]
#3: SPE-DBMS Integration

- **Goal:** Integrated querying over SPEs and traditional database systems

- **Challenges:**
  - Bridge the “data vs. operation” gap between the two worlds.
  - Find the right language and architecture primitives for the required level of querying, persistence, and performance.

- **Current state of the art:**
  - Languages [STREAM CQL, StreamSQL]
  - Architectures
    - SPE-based [typical SPEs such as Coral8, StreamBase]
    - DBMS-based [“stream-relational” systems such as TelegraphCQ/Truviso, DataCell, DejaVu, MaxStream]
MaxStream: A Platform for SPE-SPE and SPE-DBMS Integration

• Key design ideas:
  – Uniform query language and API
  – Relational database infrastructure as the basis for the federation layer (in our case: SAP MaxDB and SAP MaxDB Federator)
  – “Just enough” streaming capability inside the federation layer
MaxStream vs. Traditional Virtual Integration

<table>
<thead>
<tr>
<th>Traditional Virtual Integration</th>
<th>MaxStream</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Goal: to query across multiple autonomous &amp; heterogeneous data sources</td>
<td>• Goal: to query across multiple autonomous &amp; heterogeneous SPEs (and DBMSs)</td>
</tr>
<tr>
<td>• Source wrappers send <strong>queries</strong> and receive answers</td>
<td>• SPE wrappers send <strong>queries and data</strong>, and receive answers</td>
</tr>
<tr>
<td>• Sources host the <strong>data</strong></td>
<td>• SPEs host the <strong>CQs</strong></td>
</tr>
<tr>
<td>• Mapping between <strong>source schemas and a global schema</strong></td>
<td>• Mapping between <strong>SPE CQ models and a global CQ model</strong></td>
</tr>
<tr>
<td>• <strong>Queries are posed</strong> against the global schema</td>
<td>• <strong>CQs are posed and data is fed against the global CQ model</strong></td>
</tr>
<tr>
<td>• More focus on <strong>data locality</strong></td>
<td>• More focus on <strong>functional heterogeneity</strong></td>
</tr>
</tbody>
</table>
MaxStream Architecture
Two Key Building Blocks

• Streaming inputs through MaxStream
  – **ISTRIM operator** for persistent input events
  – Tuple queues for transient input events

• Streaming outputs through MaxStream
  – **Monitoring Select operator** over event tables
    • Persistent event tables for persistent output events
    • In-memory event tables for transient output events
Streaming Input Events

• The ISTREAM ("Insert STREAM") Operator
  – “Inspired” by the relation-to-stream operator of the same name in the STREAM Project, that streams new tuples being inserted into a given relation.
  – Example: OrdersTable(ClientId, OrderId, ProductId, Quantity)

```
INSERT INTO STREAM OrdersStream
SELECT ClientId, OrderId, ProductId, Quantity
FROM ISTREAM(OrdersTable);
```

<table>
<thead>
<tr>
<th>T</th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
</tr>
</thead>
<tbody>
<tr>
<td>T+1</td>
<td>r1</td>
<td>r2</td>
<td>r3</td>
</tr>
</tbody>
</table>

ISTREAM(OrdersTable) at $T+1$ returns: $<r4, T+1>, <r5, T+1>$
Streaming Output Events

• Opposite of streaming input events, but...
  – Unlike the SPE interface, the client application interface is not push-based.

• The Monitoring Select Operator
  – Select operation blocks until there is at least one row to return.
  – For continuous monitoring, the client program re-issues Monitoring Select in a loop.
  – Monitoring Select operates on “event tables”.

• Example: Detect unusually large order volumes.

```sql
SELECT *
FROM /*+ EVENT */ TotalSalesTable
WHERE TotalSales > 500000;
```
ISTREAM and Monitoring Select in Action

Data Feeder Client

// Continuous Insert into OrdersTable kept in MaxStream
CREATE TABLE OrdersTable;
WHILE (true) {
    INSERT INTO OrdersTable VALUES (...);
sleep(period);
};

// Continuous Insert into OrdersStream kept in the SPE
CREATE STREAM OrdersStream;
INSERT INTO STREAM OrdersStream
    SELECT ...
    FROM ISTREAM(OrdersTable);

Monitoring Clients

// Setting up the Continuous Query to push down to the SPE
INSERT INTO STREAM TotalSalesStream
    SELECT SUM(...)
    FROM OrdersStream
    KEEP 1 HOUR;

// Streaming Output Events inserted by the SPE
CREATE TABLE TotalSalesTable;
// Sales spikes Query to run in MaxStream
WHILE (true) {
    SELECT *
    FROM /*+EVENT*/ TotalSalesTable
    WHERE TotalSales > 500000;
};
Hybrid Queries in MaxStream

• Hybrid queries are continuous queries that join Streams with Tables.

• Two important factors that affect efficiency:
  – The streaming data source must be first in the join ordering.
  – Hybrid queries can be rewritten to perform the join within the MaxStream Federator, removing the need for the SPE to establish connections to external databases.

• One can conveniently use hybrid queries in MaxStream in two ways:
  – To enrich the input stream before it is passed to the SPE
  – To enrich the output stream after it is received from the SPE
MaxStream Query Model

• Problem: Heterogeneity of SPE query models
  – Syntax heterogeneity
    • Language clauses/keywords for common constructs syntactically differ.
  – Capability heterogeneity
    • Support for certain query types differs.
  – Execution model heterogeneity
    • Underlying query execution models differ.
      ➢ Not exposed to the application developer at the language syntax level.

• First step towards a solution: Create a model to analyze and predict the query execution semantics of SPEs.
The SECRET Model

- What affects query results produced by an SPE?
  - **ScopE**: Given a query with certain window properties, what are the potential *window intervals*?
  - **Content**: Given an input stream, what are the *actual contents* for those window intervals?
  - **REport**: Under what conditions, do those window contents become *visible* to the query processor for evaluation?
  - **Tick**: What drives an SPE to *take action* on a given input stream?

\[
\text{Query Result} = F(\text{system, query, input})
\]
The SECRET of a Query Plan

SECRET

ScopE

window

Tick

REport

Content

query operator

SECRET
The SECRET of an SPE

• Tick:
  – tuple-driven (e.g., Aurora, Borealis, StreamBase, TelegraphCQ, Truvisio)
  – time-driven (e.g., STREAM, Oracle CEP)
  – batch-driven (e.g., Coral8, [Jain et al, VLDB’08])

• REport:
  – window close & non-empty (e.g., StreamBase)
  – content change & non-empty (e.g., Coral8)
  – window close & content change & non-empty (e.g., STREAM)
MaxStream: Future Outlook

• Query model
  – how to extend SECRET (other query types, analysis of other SPEs, input imperfections)
  – how to use SECRET in MaxStream (SECRET-based query and SPE capability analyzer)
• Capability- and Cost-based query optimization
• Transactional stream processing
• Distributed operation
Conclusions

• Today, integration support for SPEs is needed in three main forms: across sources, SPE-SPE, SPE-DBMS.

• There are many open research challenges.

• MaxStream takes up some of these challenges for SPE-SPE and SPE-DBMS integration.

• More information about MaxStream:
  http://www.systems.ethz.ch/research/projects/maxstream/