Processing Data Streams: 
An (Incomplete) Tutorial

Johannes Gehrke 
Department of Computer Science 

johannes@cs.cornell.edu 
http://www.cs.cornell.edu

Standard Pub/Sub

- Publish/subscribe (pub/sub) is a powerful paradigm
  - Publishers generate data
    - Events, publications
  - Subscribers describe interests in publications
    - Queries, subscriptions
- Asynchronous communication
  - Decoupling of publishers and subscribers
- Much commercial software …
Limitation of Standard Pub/Sub

- Scalable implementations have very simple query languages
  - Simple predicates, comparing message attributes to constants
  - E.g., topic='politics' AND author='J. Doe'
- Individual events vs. event sequences
- Many *monitoring applications* need sequence patterns
  - Stock tickers, RSS feeds, network monitoring, sensor data monitoring, fraud detection, etc.

Example: RSS Feed Monitoring

- Once CNN.com posts an article on Technology, send me the first post referencing (i.e., containing a link to) this article from the blogs to which I subscribe
- Send postings from all blogs to which I subscribe, in which the first posting is a reference to a sensitive site XYZ, and each later posting is a reference to the previous.
Example: System Event Log Monitoring

- In the past 60 seconds, has the number of failed logins (security logs) increased by more than 5? (break-in attempt)
- Have there been any failed connections in the past 15 minutes? If yes, is the rate increasing?
- Have there been any disk errors in the past 30 minutes? If yes, is the rate increasing? (failed disk indicator)
- Have there been any critical errors (those added to the dbase table to monitor by administrators) in the past 10 minutes?

Example: Stock Monitoring

- Notify me when the price of IBM is above $83, and the first MSFT price afterwards is below $27.
- Notify me when some stock goes up by at least 5% from one transaction to the next.
- Notify me when the price of any stock increases monotonically for ≥30 min.
- Notify me when the next IBM stock is above its 52-week average.
Solutions?

- Traditional pub/sub
  - Scalable, but not expressive enough
- Database Management System
  - Static datasets
  - One-shot queries
  - Triggers
- Data Stream Management Systems
- Event Processing Systems

Real-Time DSP Requirements

1. Support a high-level “StreamSQL” language
2. Deal with out-of-order data
3. Generate predictable and repeatable outcomes
4. Integrate well with static data
5. Fault-tolerance
6. Scale with hardware resources
7. Low latency → process data as it streams by (“in-stream processing”); no requirement to store data first
Comparison of Stream Systems

<table>
<thead>
<tr>
<th>Complexity of queries</th>
<th>Number of concurrent queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Few : 😊 Publish/subscribe</td>
</tr>
<tr>
<td>High</td>
<td>Many : DSMS CEP</td>
</tr>
</tbody>
</table>

Tutorial Outline

- Basics
- How to model time
- Data stream query languages and processing models
- Fault tolerance
- New operators
- A Case Study
Temporal Model

- Questions:
  - How are timestamps defined?
  - What is the timestamp of an output record?

- Approaches:
  - Point timestamps
  - Interval timestamps

- Surprises like $E_1;(E_2;E_3)=E_2;(E_1;E_3)$?

Imperfections in Event Streaming

Slide courtesy of Mingsheng Hong.
Imperfections in Event Streaming

Network imperfections:
Tuples are late and/or out of order

Stream source retractions:
A tuple is retracted after it is streamed on the wire
Data Model

- Stream \( S \) is a sequence of tuples \( \langle \vec{a}; t_0, t_1 \rangle \)
  - \( \vec{a} = (a_1, \ldots, a_n) \) are data attribute values
    - Like relational tuples
  - \( t \)'s are temporal values
    - Starting and detection times of an event
    - Events have duration
- Example
  - Schema of stock ticker stream: (Name, Price)
Data Model

- Stream $S$ is a sequence of tuples $\langle \overline{a}, t_0, t_1 \rangle$
  - $\overline{a} = (a_1, \ldots, a_n)$ are data attribute values
  - Like relational tuples
  - $t$'s are temporal values
    - Starting and detection times of an event
    - Events have duration
  - Example
    - Schema of stock ticker stream: (Name, Price)

Cayuga Stream Algebra

- Compositional: Operators produce new streams from existing streams
- Translation to Nondeterministic Finite Automata
  - Edge transitions on input events
  - Automaton instances carry relevant data from matched events
Operators

- Relational operators (on non-temporal attributes)
  - Selection \( \sigma_\theta \)
  - Projection \( \pi_X \)
  - Renaming \( \rho_f \)
  - Union

- Together these give standard pub/sub

Sequence Operator

- **Sequence** operator \( S_1 \theta S_2 \)
- After an event from \( S_1 \) is detected, match the first event from \( S_2 \) that satisfies the condition

Examples

- IBM price increases by at least $1 in two consecutive sales:
  \[ \sigma_{p2 > p1 + 1} (\sigma_{n1 = \text{IBM}}(S_1); n2 = \text{IBM} \ S_2) \]

- Find a stock whose price stays constant in two consecutive sales:
  \[ \sigma_{p1 = p2} (S_1; n1 = n2 \ S_2) \]
Sequence Operator (Contd.)

- Sequencing is a weak join on timestamps
  - Can join an event with one later in future...
  - Or with the immediate successor
    - Can be useful for queries about causal relationships

Automaton for $\rho_f \circ \sigma_{\theta_2} (\mathcal{E}_1 ; \theta_1 S)$

Sequence Operator: Example

- Query 1:
  - Send me the first new posting from apple.slashdot.org after a product announcement on www.apple.com.
Sequence Operator (Contd.)

- Automaton edges search for matches.
  - $\theta_1$: www.apple.com announcement
  - $\theta_2$: apple.slashdot.org posting
- Intermediate state stores Apple announcements
  - Waits to pair with next available Slashdot post.

Parameterized Sequencing

- Problems with previous query
  - Assumes a quick response to Apple announcements
  - There may be several announcements (i.e., MacWorld Expo)
- Want Slashdot post to refer to right product
  - Post has link to announcement as a parameter
- Query 2:
  - Once a new product announcement appears on www.apple.com, send me the first posting from apple.slashdot.org that links to this announcement.
Parameterized Sequencing (Contd.)

- Intermediate information is already there
  - Each announcement is an automaton instance
- Just change edge filters to leverage information
  - $\theta_1$: www.apple.com announcement
  - $\theta_2$: apple.slashdot.org posting linking to an instance

Iteration Operator

- *Iteration* operator (similar to Kleene-+)

\[ \mu_{\mathcal{A}, \theta}(S_1, S_2) \]

- Intuitively:

\[ \mathcal{A}(S_1 \circ_{\theta} S_2) \cup \mathcal{A}(\mathcal{A}(S_1 \circ_{\theta} S_2) \circ_{\theta} S_2) \cup \cdots \]
### Iteration Example

- IBM stock price monotonically increases

\[
\mu_{p_{2 > p_{2 \text{last}}, n2 = \text{IBM}}} (\sigma_{n1 = \text{IBM}}(S_1), S_2)
\]

#### Automaton for Iteration Operator

Automaton for \( \rho_{f_2} \circ \sigma_{\theta_3} (\mu_{\rho_{f_1} \circ \sigma_{\theta_2}, \theta_1} (E_1, S')) \)
Iteration: Another Example

- Following the spread of crazy Apple rumors...

Query 3:
- Send me a sequence of Apple blog postings, in which the first posting is a rumor about an upcoming Apple product announcement, and each later posting is a reference (i.e., contains a direct quote from or a hypertext link to) to the previous.

Implementing Iteration

- Similar to parameters sequencing
  - $\theta_1$: Initial Apple rumor
  - $\theta_2$: Rumor that references the previous one
- Purple edge is a rebind edge
  - Updates instance information with latest rumor
Aggregation

- Recall: Iteration also allows for aggregation.
  - Iterate over all posts of this type
  - Keep a running aggregate of some post attribute
    - e.g. current number of comments, average word count, etc...
  - Implemented like normal aggregates
    - Need initializer, iterator, finalizer

- Query 4:
  - Send me an product review from apple.slashdot.org once it receives an above average number of user comments.

Implementing Aggregation

- Rebind edge performs the aggregation
  - $g$ is attached to rebind edge to update values
- Note outgoing edge different from rebind edge
  - $\theta_3$: Above average number of comments
Other Features

- Resubscription
  - Ability for one query to subscribe to the output of another (as a stream)
  - Significantly more expressive

- Extensibility
  - Incorporate user-defined datatypes, data mining algorithms, predicates, aggregation functions, ...

Example

- Notify me when
  1. for any stock, there is a very large trade (volume > 10K);
  2. followed by a monotonic decrease in price for at least 10 minutes;
  3. the next quote on the same stock after this monotonic sequence is 5% above the previously seen (bottom) price.

- Intuition: Large sale, followed by price drop, followed by sudden upwards move
Example

- **Algebra expression:**  
  \[ \sigma_{\theta_5}(\sigma_{\theta_4}(\mu_{\sigma_{\theta_3}, \theta_2}(S_1, S_2)) \land \theta_2) S_3) \]

  \[ S_1 \equiv \rho_{\theta_1} \circ \pi_{\text{name,price}} \circ \sigma_{\theta_1}(S) \]
  
  \[ S_2 \equiv \rho_{\theta_2} \circ \pi_{\text{name,price}}(S) \]
  
  \[ S_3 \equiv \rho_{\theta_3} \circ \pi_{\text{name,price}}(S) \]

  \[ \theta_1 \equiv \text{vol} > 10,000 \]
  
  \[ \theta_2 \equiv \text{company} = \text{company.last} \]
  
  \[ \theta_3 \equiv \theta_2 \land \text{minP} < \text{minP.last} \]
  
  \[ \theta_4 \equiv \theta_3 \land \text{DUR} \geq 10 \min \]
  
  \[ \theta_5 \equiv \theta_2 \land \text{price} > 1.05 \text{minP} \]

  \[ f_1 \equiv (\text{name,price}) \mapsto (\text{company, maxP}) \]
  
  \[ f_2 \equiv (\text{name,price}) \mapsto (\text{company, minP}) \]
  
  \[ f_3 \equiv (\text{name,price}) \mapsto (\text{company, finalP}) \]

---

Example

- **Diagram:**

\[ \theta_1 \equiv \text{vol} > 10,000 \]
  
  \[ \theta_2 \equiv \text{company} = \text{company.last} \]
  
  \[ \theta_3 \equiv \theta_2 \land \text{minP} < \text{minP.last} \]
  
  \[ \theta_4 \equiv \theta_3 \land \text{DUR} \geq 10 \min \]
  
  \[ \theta_5 \equiv \theta_2 \land \text{price} > 1.05 \text{minP} \]

  \[ f_1 \equiv (\text{name,price}) \mapsto (\text{company, maxP}) \]
  
  \[ f_2 \equiv (\text{name,price}) \mapsto (\text{company, minP}) \]
  
  \[ f_3 \equiv (\text{name,price}) \mapsto (\text{company, finalP}) \]
Example

\[(\text{name, price, vol})\]  
\[(\text{company, maxP})\]  
\[(\text{company, maxP, minP, finalP})\]

\[\theta_1 \equiv \text{vol} > 10,000\]
\[\theta_2 \equiv \text{company} = \text{company}.last\]
\[\theta_3 \equiv \theta_2 \wedge \text{minP} < \text{minP}.last\]
\[\theta_4 \equiv \theta_3 \wedge \text{DUR} \geq 10\text{ min}\]
\[\theta_5 \equiv \theta_2 \wedge \text{price} > 1.05\text{ minP}\]
\[f_1 \equiv (\text{name, price}) \mapsto (\text{company, maxP})\]
\[f_2 \equiv (\text{name, price}) \mapsto (\text{company, minP})\]
\[f_3 \equiv (\text{name, price}) \mapsto (\text{company, finalP})\]

Example

\[(\text{name, price, vol})\]  
\[(\text{company, maxP, minP, finalP})\]

<table>
<thead>
<tr>
<th>Input event</th>
<th>Instances at state A</th>
<th>Instances at state B</th>
<th>Instances at state C</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_1 : (BM, 90, 15000; 910, 910)</td>
<td>(I_1 = (\text{BM}, 90, \text{NULL}; 910, 910))</td>
<td>(I_2 = (\text{BM}, 90, \text{NULL}; 910, 910))</td>
<td>(I_3 = (\text{BM}, 90, \text{NULL}; 910, 910))</td>
</tr>
<tr>
<td>e_2 : (BM, 95, 7000; 915, 915)</td>
<td>(I_1 = (\text{BM}, 95, \text{NULL}; 915, 915))</td>
<td>(I_2 = (\text{BM}, 95, \text{NULL}; 915, 915))</td>
<td>(I_3 = (\text{BM}, 95, \text{NULL}; 915, 915))</td>
</tr>
<tr>
<td>e_3 : (hll, 90, 11000; 917, 917)</td>
<td>(I_1 = (\text{hll}, 90, \text{NULL}; 917, 917))</td>
<td>(I_2 = (\text{hll}, 90, \text{NULL}; 917, 917))</td>
<td>(I_3 = (\text{hll}, 90, \text{NULL}; 917, 917))</td>
</tr>
<tr>
<td>e_4 : (BM, 81, 8000; 921, 921)</td>
<td>(I_1 = (\text{BM}, 81, \text{NULL}; 921, 921))</td>
<td>(I_2 = (\text{BM}, 81, \text{NULL}; 921, 921))</td>
<td>(I_3 = (\text{BM}, 81, \text{NULL}; 921, 921))</td>
</tr>
<tr>
<td>e_5 : (MSFT, 25, 6000; 923, 923)</td>
<td>(I_1 = (\text{MSFT}, 25, \text{NULL}; 923, 923))</td>
<td>(I_2 = (\text{MSFT}, 25, \text{NULL}; 923, 923))</td>
<td>(I_3 = (\text{MSFT}, 25, \text{NULL}; 923, 923))</td>
</tr>
<tr>
<td>e_6 : (BM, 91, 9000; 924, 924)</td>
<td>(I_1 = (\text{BM}, 91, \text{NULL}; 924, 924))</td>
<td>(I_2 = (\text{BM}, 91, \text{NULL}; 924, 924))</td>
<td>(I_3 = (\text{BM}, 91, \text{NULL}; 924, 924))</td>
</tr>
</tbody>
</table>
Cayuga Query Language

```sql
SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(  
    FILTER{DUR > 10min}(  
      (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice  
      FROM FILTER{Volume > 10000}(Stock))  
    FOLD{$2.Name = $.Name, $2.Price < $.Price}  
    Stock)  
  NEXT{$2.Name = $1.Name}  
  Stock)
```

Cayuga Automata

```sql
SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(  
    FILTER{DUR > 10min}(  
      (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice  
      FROM FILTER{Volume > 10000}(Stock))  
    FOLD{$2.Name = $.Name, $2.Price < $.Price}  
    Stock)  
  NEXT{$2.Name = $1.Name}  
  Stock)
```
SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
FILTER{Price > 1.05*MinPrice}(
  FILTER{DUR > 10min}(
    (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
     FROM FILTER{Volume > 10000}(Stock))
    FOLD{$2.Name = $.Name, $2.Price < $.Price}
    Stock)
  NEXT{$2.Name = $1.Name}
  Stock)

Cayuga Automata

SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
FILTER{Price > 1.05*MinPrice}(
  FILTER{DUR > 10min}(
    (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
     FROM FILTER{Volume > 10000}(Stock))
    FOLD{$2.Name = $.Name, $2.Price < $.Price}
    Stock)
  NEXT{$2.Name = $1.Name}
  Stock)
Cayuga Automata

SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
FILTER{Price > 1.05*MinPrice}(
  FILTER{DUR > 10min}(
    (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
     FROM FILTER{Volume > 10000}(Stock))
    FOLD{$2.Name = $.Name, $2.Price < $.Price}
    Stock)
  NEXT{$2.Name = $1.Name}
  Stock)

Cayuga Automata

SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
FILTER{Price > 1.05*MinPrice}(
  FILTER{DUR > 10min}(
    (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
     FROM FILTER{Volume > 10000}(Stock))
    FOLD{$2.Name = $.Name, $2.Price < $.Price}
    Stock)
  NEXT{$2.Name = $1.Name}
  Stock)
Cayuga Automata

```
SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(
    FILTER{DUR > 10min}(
      (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
       FROM FILTER{Volume > 10000}(Stock))
      FOLD{$2.Name = $.Name, $2.Price < $.Price}
      Stock)
    NEXT{$2.Name = $1.Name}
    Stock)
```

Example: Double-Top

- Double-Top query pattern
Cayuga Resubscription (No Iteration)

- Compute stream of local extrema: \( \sigma_{\theta_1}(\sigma_{\theta_2}(S_1 ; S_2) ; S_3) \)
- Union them, then search for actual pattern:

\[
\sigma_{\theta_1}(\sigma_{\theta_2}(\sigma_{\theta_3}(E_1 ; E_2) ; E_3) ; E_4) ; E_5)
\]

\[
\begin{align*}
\theta_1 & \equiv (E_5.\text{price} \leq E_1.\text{price}) \\
\theta_2 & \equiv (0.9E_2.\text{price} \leq E_4.\text{price} \leq 1.1E_2.\text{price}) \\
\theta_3 & \equiv (0.9E_1.\text{price} \leq E_3.\text{price} \leq 1.1E_1.\text{price}) \\
\theta_4 & \equiv (E_2.\text{price} \geq 1.2 \times E_1.\text{price}) \\
\theta_5 & \equiv (E_2.\text{name} = E_1.\text{name}) \\
\theta_6 & \equiv (E_3.\text{name} = E_1.\text{name}) \\
\theta_7 & \equiv (E_4.\text{name} = E_1.\text{name}) \\
\theta_8 & \equiv (E_5.\text{name} = E_1.\text{name})
\end{align*}
\]

Double-Top Query: Cayuga

```
FROM FILTER (Price >= Price_1 AND Price <= PriceA)
    (FILTER (Price <= 1.1*PriceB) 
        SELECT Name, PriceA, PriceB, PriceC, Price_1 AS PriceD, Price
        FROM
        FILTER (Price <= 0.9*PriceB) 
            SELECT Name, PriceA, PriceB, Price_1 AS PriceD, Price
            FROM
        FILTER (Price <= 0.9*PriceA AND Price <= 1.1*PriceA) 
            SELECT Name, PriceA, Price_1 AS PriceD, Price
            FROM
        FILTER (Price <= 1.2*PriceA) 
            SELECT Name, Price_1 AS PriceA, Price
            FROM
        FILTER (Price < Price_1) 
            SELECT Name, Price FROM Stock NEXT ($1.Name=$2.Name) Stock
        FOLD ($1.Name = $2.Name, $2.Price >= $.Price) Stock
        FOLD ($1.Name = $2.Name, $2.Price <= $.Price) Stock
        FOLD ($1.Name = $2.Name, $2.Price >= $.Price) Stock
        FOLD ($1.Name = $2.Name, $2.Price <= $.Price) Stock
        NEXT ($1.Name = $2.Name)
    )
PUBLISH MShapeStock
```
Double-Top Query: CQL

query : Rstream (Select S.time, S.name, S.price, (S.price - P.price) From Stock [Now] as S, Stock [Partition By P.name Rows 2] as P Where S.name = P.name and S.time > P.time);
vtable : register stream StockDiff (time integer, name integer, price float, pdiff float);

query : Rstream (Select P.time, P.name, P.price, P.pdiff From StockDiff [Now] as S, StockDiff [Partition By P.name Rows 2] as P Where S.name = P.name and (S.pdiff * P.pdiff) < 0.0);
vtable : register stream Extrema (time integer, name integer, price float, pdiff float);

query : Select name, count(*) from Extrema Group By name;
vtable : register relation ExtremaCounter (name integer, seqNo integer);

query : Rstream (Select E.name, E.price, E.pdiff, C.seqNo, C.seq.No – 1 From Extrema [Now] as E, ExtremaCounter as C Where E.name = C.name);
vtable : register stream ExtremaSeq (name integer, price float, pdiff float, seq integer, prevSeq integer);

query : Select name, price, seq from ExtremaSeq Where pdiff < 0.0;
vtable : register relation stateA (name integer, price float, seq integer);

query : Rstream (Select E.name, E.price, B.price, B.aprice, E.seq From ExtremaSeq [Now] as E, stateA as A Where E.name = A.name and E.prevSeq = A.seq and E.price > (A.price * 1.2));
vtable : register relation stateB (name integer, bprice float, aprice float, seq integer);

query : Rstream (Select E.name, E.price, B.bprice, B.aprice, E.seq From ExtremaSeq [Now] as E, stateB as B Where E.name = B.name and E.prevSeq = B.seq and E.price > (B.bprice * 0.9) and E.price < (B.aprice * 1.1));
vtable : register relation stateC (name integer, cprice float, bprice float, aprice float, seq integer);

query : Rstream (Select E.name, E.price, C.cprice, C.bprice, C.aprice, E.seq From ExtremaSeq [Now] as E, stateC as C Where E.name = C.name and E.prevSeq = C.seq and E.price > (C.bprice * 0.9) and E.price < (C.aprice * 1.1));
vtable : register relation stateD (name integer, dprice float, cprice float, bprice float, aprice float, seq integer);

query : Rstream (Select E.name, E.price, D.dprice, D.cprice, D.bprice, D.aprice, E.seq From ExtremaSeq [Now] as E, stateD as D Where E.name = D.name and E.prevSeq = D.seq and E.price <= D.aprice);

Example: Double-Top

- Real stock data (24 companies, 112,635 events)