A Model for Continuous Query Latencies in Data Streams

R. Baldoni\textsuperscript{\circ}  G. Di Luna\textsuperscript{\circ}  D. Firmani\textsuperscript{\circ}  G. Lodi\textsuperscript{\circ}

\textsuperscript{\circ} Sapienza, University of Rome

September 19, 2011
Introduction

Problem Statement

Model

Example

Conclusions and Future Works
Data Streams Query Processing

In recent years we have witnessed an increased adoption of Data Streams Query Processing in several application domains.

▶ Data Base Management Systems
  ▶ Mostly static data, ad-hoc one-time queries
  ▶ Fire the queries at the data, return result sets

▶ Data Stream Management Systems / Complex Event Processing Systems
  ▶ Mostly transient data, continuous queries
  ▶ Fire the data at the queries, incrementally update result streams
At a very high level, a programmer, in order to solve a continuous query:

- defines a set of functions;
- describes how incoming flows of information, i.e. data streams or events, have to be processed to timely produce the target stream as output;
- by producing intermediate streams useful to the computation.
Application Domains

- **Financial analysis**: Algorithmic trading: detect advantageous market conditions, automatically execute trades;
- **Network Monitoring**: Intrusion detection;
- **Fraud Detection**;
- **Sensor Networks**: Health care, Habitat monitoring.

Latency is fundamental.
Related Works

A lot of work has been done to minimize latency on DSMS:

▶ optimized query evaluation planning;
▶ avoiding overload of operators in distributed environments;
▶ resilient operator placement;
▶ ...

Our focus is to propose a cost evaluation technique to estimate whether a given strategy to solve a query fits the QoS requirements, independently from the used DSMS and before an experimental validation phase.
How can we evaluate latency?

QoS requirements from latency point of view:

- time cost to produce a new output stream item;
- rate of the output stream;
- possibility to improve the time needed to trigger a solution.

Our approach is to compare different **data-flow graphs** in a platform independent framework.
Data Flow Graph: Definition

▶ **EPU.** An Event Processing Unit is a function that takes streams as input, performs a computation and originates a single stream as output for downstream consumption. An EPU can be:
  ▶ a relational operator (e.g., Esper);
  ▶ any user-defined operator (e.g., Spade).

▶ **DFG.** A data flow graph, that represents a strategy to solve a query, is a DAG $G = (V, E)$ s.t.
  ▶ $V$ contains all the EPU nodes needed for the computation;
  ▶ in $E$ there exists an edge $(v, u)$ iff there exists an EPU $v \in V$ that produces an event stream which is consumed by an EPU $u \in V$. 
**Data Flow Graph: Example**

**Query:** Process a raw *market data feed* and detect when the data rate of a feed falls off unexpectedly, in order to alert when there is a possible problem with the feed.

<table>
<thead>
<tr>
<th>EPU</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>String symbol; FeedEnum feed; double bidPrice; double askPrice;</td>
</tr>
<tr>
<td>$u_2$</td>
<td>insert into TicksPerSecond select feed, count(*) as cnt from MarketDataEvent.win:time_batch(1 second) group by feed</td>
</tr>
<tr>
<td>$u_3$</td>
<td>select feed, avg(cnt) as avgCnt, cnt as feedCnt from TicksPerSecond.win:time(10 seconds) group by feed having cnt &lt; avg(cnt) * 0.75</td>
</tr>
</tbody>
</table>
Metrics

Given a data-flow graph $G$ and a set of input streams $S$ that produces an output stream, compute:

- **Output Lat**: begin of the input $\rightarrow$ begin of the output streams
- **Activity Lat**: begin of the input $\rightarrow$ end of the output streams
- **Reactivity Lat**: end of the input $\rightarrow$ begin of the output streams
- **Complexity**: minimum dimension of the input streams necessary to produce a non-empty output stream
Model Abstraction

EPU behavior:

ASB/O  All-Streams Batch/Online Processing (logical and/or);

EB/TB  Event/Time Based (detect fall-off/ticks per sec)

EPU parameters:

$t_u(v)$  time window of a TB $u$ wrt $v$ that produces the $u$ input;

$n_u(v)$  average number of events that a EB $u$ consumes from $v$ in order to produce a single event;

$n(u)$  dimension of co-domain of the function computed by the $u$;

$p(u)$  average time in which $u$ computes the function (update output stream).
Evaluating EPU Metrics

Evaluation of the *DFG metrics* on the basis of evaluation of the metrics that each EPU has wrt its input consumption and output production.

EPU metric evaluation performed by computing:

- input and output rate $\rho_u(*)$, $\rho(u)$;
- input and output silence period $\sigma_u(*)$, $\sigma(u)$. 
Evaluating DFG Metrics

Algorithms for computing DFG metrics.

- evaluate EPU metrics following any topological sort of data-flow graph $G$;
- algorithm for a metric $M(G)$ consists in a graph visit that finds the $M$-critical path, i.e., the set of EPU that determines the final value of $M(G)$.
A simple Example

Back to *market data feed* example.

<table>
<thead>
<tr>
<th></th>
<th>$l(u)$</th>
<th>$n_u(*)$</th>
<th>$t_u(*)$</th>
<th>$n(u)$</th>
<th>$p(u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>${}$</td>
<td>${}$</td>
<td>${}$</td>
<td>1</td>
<td>$y_1$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>${u_1}$</td>
<td>-</td>
<td>${1}$</td>
<td>1</td>
<td>$y_2$</td>
</tr>
<tr>
<td>$u_3$</td>
<td>${u_2}$</td>
<td>${x_3}$</td>
<td>-</td>
<td>1</td>
<td>$y_3$</td>
</tr>
</tbody>
</table>

$x_3 \in [1, \infty)$

Reactivity Latency evaluation:

- Let us consider that at $t_i$ in the marked data stream we have a fall-off, and at $t_o$ the strategy represented by the DFG effectively detects it;

- In the performed abstraction we obtain that the relationship between $t_i$ and $t_o$ is given by the sum of processing times of the EPUs and the timing of the TB EPU.
The critical path changes as a function of the EPU parameters;

- Metrics may be complex functions, difficult to compute and study.

At this end:

- We implemented a software tool to handle complex DFGs;

- Experimentally evaluation of the model is still in progress.
Conclusions

- We propose a formal model to evaluate some cost metrics of a continuous streaming computation, represented as a data-flow query graph where each node is a basic query (EPU).

- The model is able to associate several metrics with a data-flow in order to evaluate the expected latency before its effective implementation on a DSMS.
THANKS FOR YOUR ATTENTION.

😊

QUESTIONS?